Universidad Carlos III de Madrid

Escuela Politécnica Superior

Ingeniería Técnica en Informática de Gestión

Proyecto Fin de Carrera

Desarrollo de un módulo basado en NopCommerce para la creación automatizada de librerías *online*

Autor: D. Ignacio Vicente Dorado

Tutor: Prof. D. Fernando Fernández Rebollo

Octubre de 2015

"*La ignorancia afirma o niega rotundamente; la ciencia duda.*"
Voltaire

Agradecimientos

En la realización de este proyecto culmina mi período en la Universidad que siempre me traerá algunos de los mejores recuerdos de mi vida. Ya entonces conocía a todas las personas que me han apoyado para cerrar por fin esta etapa como es necesario, con un proyecto Fin de Carrera que materializa el título de Ingeniero.

Es probable que sin el ánimo de las personas a las que me refiero, no habría hecho el tremendo esfuerzo que ha supuesto este trabajo, o en algún punto del mismo habría tirado la toalla. Esas personas son mi mujer, Marta, sin cuya completa y admirable predisposición de principio a fin esto no habría sido posible con toda seguridad. Y mis padres, Araceli y Juan Carlos por su constante preocupación e interés en que esto saliera adelante. Gracias de todo corazón.

He de agradecer también la ayuda desinteresada de mis dos hijos, Inés y David, a quienes dedico este proyecto, porque ya sólo estando a mi lado y respetando los momentos de máxima concentración, como no es natural en niños pequeños, me han proporcionado buena parte de la fuerza necesaria para seguir empujando.

Muchas gracias a mis hermanos Carlos, María y Paula, por estar siempre ahí y a mis amigos por echar una mano con los momentos de necesaria desconexión.

Agradezco también la ayuda que me ha prestado mi tutor, Fernando, sobre todo orientándome en el aspecto más académico y teórico del trabajo.

Índice general

Índice de figuras

Índice de tablas

Acrónimos

ACL: *Access Control List*. Sistema de asignación de permisos y privilegios a los usuarios de un software, en función del rol atribuido.

AJAX: *Asynchronous Javascript and XML*. Conjunto de técnicas de desarrollo web utilizando Javascript y XML para crear aplicaciones interactivas.

API: *Application Programming Interface*. Es el conjunto de subrutinas, funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción. [33]

ASP: *Active Server Pages*. Tecnología para desarrollo de aplicaciones web incrustado en HTML utilizando el lenguaje de programación Visual Basic Script, de Microsoft.

BBDD: Base de datos

BISAC: *Book Industry Standards and Communications.*

COM: *Component Object Model*. Tecnología de comunicación entre aplicaciones de Microsoft.

CRM: *Contact Relationship Management*. Aplicación de gestión de información para sistematizar la relación con los contactos en una compañía.

EAN: *European Article Number*. Identificador único de artículos a nivel europeo.

ERP: *Enterprise Resource Planning*. Aplicación de gestión de recursos humanos y planificación, aunque hoy en día abarcan muchas más áreas.

GUID: *Globally Unique Identifier*. Identificador único utilizado en programación y bases de datos.

HTML: *Hypertext Mark-up Language*. Lenguaje de marcado para definición del aspecto de páginas web, destinado a su procesado por parte de los navegadores.

HTTP: *Hypertext Transfer Protocol*. Protocolo de comunicaciones para redes definido por *World Wide Web Consortium*.

HTTPS: *Hypertext Transfer Protocol Sercure*. Modalidad segura de protocolo HTTP.

IIS: *Internet Information Server*. Servidor web de Microsoft.

ISBN: *International Standard Book Number*. Identificador único para libros de uso internacional.

JSON: *Javascript Object Notation*. Estructura de datos utilizada en programación de Javascript.

JSP: *Java Server Pages*. Tecnología para desarrollo de aplicaciones web incrustado en HTML utilizando el lenguaje de programación Java, de Sun Microsystems.

MVC: *Model View Controller*. Patrón de desarrollo que utiliza el paradigma de programación de modelo-vista-controlador.

OAuth: *Open Authorization*. Protocolo de autenticación para Internet, principalmente web-services.

PDF: *Portable Document Format*. Formato de documento digital universalmente aceptado.

PHP: *PHP Hypertext Preprocessor*. Lenguaje de programación para servidores web, que se incrusta en HTML.

RPC: *Remote Procedure Call*. Protocolo de red que permite a un programa de computadora ejecutar código en otra máquina remota sin tener que preocuparse por las comunicaciones entre ambas. [35]

SOAP: *Simple Object Access Protocol*. Definición de la comunicación entre dos aplicaciones por medio de intercambio de datos XML. [36]

SQL: *Structured Query Language*. Lenguaje de programación para la gestión de bases de datos.

SSL: *Secure Sockets Layer*. Protocolo criptográfico para establecer comunicaciones seguras en una red.

URL: *Universal Resource Location*. Es un identificador de recursos uniforme (*Uniform Resource Identifier*, URI) cuyos recursos referidos pueden cambiar, esto es, la dirección puede apuntar a recursos variables en el tiempo. [34]

WCF: *Windows Communication Foundation. Framework* de desarrollo de aplicaciones.

1. Introducción

Este apartado presenta al lector este documento explicando, de manera breve, la motivación que me llevó a realizar este proyecto, los objetivos que se han perseguido y, por último, la estructura del documento de manera que conozca rápidamente qué tema se trata, por qué, y cómo se va a enunciar.

1.1. Motivación

Las tecnologías *world wide web* han evolucionado muy rápidamente desde la década de los 90 cuando surgieron, hasta nuestros días, cuando ya se utilizan como mucho más que una simple fuente de información como, por ejemplo, venta a través de Internet. Los productos de software que pueden englobarse dentro de los llamados *e-commerce*, son más avanzados cada día que pasa, utilizando tecnologías y *frameworks* que facilitan al desarrollador hacer interfaces y funcionalidades más ricas, tanto para el vendedor como para el comprador.

Debido a que el interés por este tipo de software existe desde que el autor de este proyecto comenzó su carrera académica, siempre tuvo en mente hacer un proyecto que se basara en un *e-commerce*, aplicando los conocimientos adquiridos durante su paso por la Universidad, e intentar aportar sus ideas y motivaciones tanto a la comunidad abierta de desarrolladores, como a cualquier futuro alumno que pretenda continuar su trabajo, o bien basarse en él para hacer un proyecto similar.

De tal manera, se pretende que el presente documento sea también una guía y referencia sobre cómo facilitar el trabajo a los gestores y administradores web, especialmente de *e-commerce*, a la hora de poner en marcha tiendas *online*, en el caso que nos ocupa, de venta de libros. Se aplican, para ello, fundamentos de la programación existentes a día de hoy, pero apoyándose en paradigmas ya conocidos, y que se siguen utilizando por los fabricantes para crear *frameworks* de desarrollo porque se ha comprobado su utilidad a la hora de mejorar la productividad de cualquier fase del ciclo de vida del software.

Hay un problema generalizado que nos encontramos en los *e-commerce* actualmente. Por norma general, la gestión del alta de productos, cuando estos deben ser incorporados a la base de datos, es lenta y laboriosa, ya que requiere de la introducción de un sinfín de datos y atributos del producto manualmente, así como imágenes, su categorización, etc.

La solución que plantea el presente proyecto, se sirve de repositorios de datos ya existentes y que están disponibles a través de una API, para sacar partido a toda esa información de manera que sea posible incorporarlos de manera automatizada a nuestro *e-commerce*, ahorrando así un tiempo muy valioso a los administradores del sitio. Yendo más allá, también se plantea la creación de sitios web automáticamente, basándonos en el concepto anteriormente enunciado, y así dar la posibilidad a, por ejemplo, vendedores al por mayor (*wholesalers*) de dar un servicio avanzado y personalizado a sus distribuidores proporcionándoles una tienda *online* con sólo un clic.

1.2. Objetivos

La idea central de este proyecto fin de carrera, nace tras plantearse la cuestión de cómo aumentar la eficiencia a la hora de incorporar información en los *e-commerce* existentes a día de hoy.

Como objetivo principal se propone la creación automatizada de webs *e-commerce* cargando los productos desde repositorios de datos remotos servidos por un servicio web proporcionado por terceros. En el caso que exponemos, Google Books[1] es el servicio de terceros que facilita los datos de los que se nutre el programa para generar dichas tiendas. Dicho servicio cuenta con una librería de funciones y clases que se incorporan a la solución del presente proyecto, para mayor facilidad al programar.

Además, se persiguen otros objetivos también importantes, como son:

- poner de relieve la importancia de los *e-commerce* en la sociedad actual mediante el estudio de su trayectoria y situación actual y analizar las posibilidades que existen a día de hoy en función de su tipo de licenciamiento
- la búsqueda de soluciones al problema habitual de introducir gran cantidad de productos en el sistema
- el diseño del prototipo de una extensión que se encargue de obtener los datos y propiedades de libros para hacer automática la adición de los mismos a un *e-commerce*, de manera que cualquier ingeniero de software pueda comprender y aplicar a cualquier otro sistema
- la implementación, que se basa en multitud de tecnologías que se detallan a lo largo de esta memoria y que gestionan la comunicación entre NopCommerce y Google Books API
- probar el sistema propuesto y comprobar que efectivamente es funcional y soluciona el problema expuesto.

[1] https://books.google.es/

1.3. Contenido de la memoria

Este apartado detalla cómo está estructurado el presente documento, para una mejor comprensión y poder hacer una lectura más ordenada.

1. Introducción

En "Introducción" se trata de enunciar cuáles han sido las motivaciones que han llevado a realizar este proyecto, qué objetivos se perseguían y qué se puede encontrar en el resto del documento.

2. Estado de la cuestión

En "El estado de la cuestión", se habla sobre la historia del *e-commerce*, se detalla qué es NopCommerce, su evolución, qué tecnologías utiliza y qué alternativas existen a este software. También cómo surge Google Books, el servicio del que nos hemos servido para obtener datos fiables de libros y qué otros servicios se evaluaron y por qué se decidió por Google Books. Además, se incluye una breve explicación de qué es BISAC[2], que es el estándar documental utilizado por Google Books para categorizar sus obras, para qué se utiliza en este trabajo y qué cubre dicho estándar. Por último, se enuncia qué es el módulo SignalR utilizado para envío de mensajes de manera asíncrona entre el servidor y el navegador.

3. Análisis

En "Análisis" se habla sobre:

- **Requisitos del sistema:** se especifica qué necesidad debe cubrir el aplicativo desde la perspectiva del usuario final, en términos de rendimiento y funcionalidad. Se categorizan en funcionales (directamente relacionados con lo que debe hacer el programa) y no funcionales (indirectamente relacionados).
- **Casos de uso:** se analizan las principales interacciones entre los administradores de NopCommerce, los administradores web, los usuarios y el sistema.

[2] https://www.bisg.org/bisac-subject-codes

- **Diagrama de clases:** se muestra el diagrama de las clases más importantes que intervienen en el desarrollo de la presente aplicación.
- **Diagrama de flujo:** se representa el flujo principal de uso de la extensión, mediante un diagrama.

4. Diseño

En "Diseño e implementación" se abarca la arquitectura del sistema y el modelado Entidad/Relación en el que nos basamos para construir el modelo de datos del patrón MVC. Dicho patrón, se basa en el paradigma de Modelo-vista-controlador, que divide conceptualmente el software en datos, lógica de negocio y presentación para así facilitar la programación de cada uno de los componentes y su mantenimiento.

5. Implementación

Se muestra cómo se organiza la solución dentro del proyecto, se especifica en detalle cómo se desarrolla cada uno de los módulos MVC, con ejemplos de código entre explicaciones y comentarios.

6. Pruebas

En el apartado de "Pruebas" se detalla qué *tests* se deben llevar a cabo en la aplicación para asegurar el correcto flujo de información y una adecuada experiencia de usuario.

7. Conclusiones

En "Conclusiones" se habla sobre el resultado final del proyecto, qué beneficios reporta a los administradores NopCommerce y los administradores de tiendas de libros.

8. Trabajo futuro

En este apartado se habla de qué proyectos podrían realizarse partiendo de la base del presente trabajo, y qué ideas surgen durante el desarrollo para ser acometidos en una hipotética continuación.

9. Referencias y bibliografía

El apartado de "Referencias y bibliografía" recoge todas las fuentes consultadas tanto en Internet como de carácter bibliográfico para completar el desarrollo del aplicativo objeto de este documento.

10. Anexos

El capítulo "Anexos" reúne toda la documentación adicional que generalmente tiene como finalidad ser consultada mientras se lee esta memoria. En dicha sección se incluye todos los datos de gestión del proyecto, y manuales de usuario y de instalación.

2. Estado de la cuestión

En este capítulo se analiza la historia y el momento que atraviesan el *e-commerce*, nopCommerce, plataforma en la que se basa la extensión objeto del proyecto, Google Books y el estándar BISAC.

2.1. Breve historia del *e-commerce*

Hasta los años 90, momento en que comenzó la utilización de las redes interconectadas tal y como lo conocemos hoy, el *World Wide Web*, los intercambios de información que se hacían entre equipos con fines comerciales, no habían estado nunca orientados al consumo minorista. Cabe destacar en los inicios el sistema "EDI", el cual se sigue utilizando en nuestros días, pero que se limitaba, y actualmente se limita, al intercambio de información transaccional entre compañías.

Con la llegada del acceso masivo a Internet, varias fueron las compañías que vieron una oportunidad en la venta *online* y se lanzaron rápidamente a desarrollar el negocio. Buenos ejemplos de este hecho son Amazon o eBay que nacieron en 1995 (ver Figura 1) y que también mantienen su actividad a día de hoy tras un crecimiento muy significativo. Los organismos internacionales también reaccionaron rápidamente y comenzaron a regular este mercado, principalmente, con el objetivo de facilitar las transacciones entre empresas a nivel mundial y proveerlas de seguridad jurídica.

Figura 1. Amazon en 1997. Fuente: http://blog.paulinepauline.de/2007/10/03/das-ende-der-amazon-reiternavigation/

Hitos importantes en la carrera del *e-commerce* fueron, por ejemplo:

- En 1997, Dell[3] supera el millón de dólares por venta *online*.
- En 2002, eBay[4] compra el sistema de pagos *online* PayPal[5] lo que le da gran popularidad y pasa a convertirse en el sistema de pagos *online* de referencia en todo el mundo, gestionando un gran volumen de operaciones por su versatilidad y fácil instalación en cualquier *e-commerce*.
- En 2007, Apple[6] lanza el iPhone.
- En 2008, Google [7] lanza Android[8].
- Estos dos lanzamientos (ver figuras 2 y 3) son importantes para entender el crecimiento del *e-commerce* ya que, estos dos fabricantes, generalizan el acceso al smartphone, lo que poco a poco provoca que la compra *online* comience a tener presencia en el mundo del móvil por la comodidad y

[3] Dell - www.dell.es
[4] eBay - www.ebay.es
[5] PayPal - www.paypal.com/es/home
[6] Apple - www.apple.com
[7] Google - www.google.com
[8] Android - www.android.com

disponibilidad que proporcionan. Las interfaces empiezan a diseñarse específicamente para pantallas de 4 y 5 pulgadas, y se diseñan aplicaciones para el móvil destinadas a comprar.

Figura 2. Primer iPhone. Fuente: http://www.theverge.com/2014/9/9/6125849/iphone-history-pictures

Figura 3. Primer Android. Fuente: http://www.cyberhades.com/2012/12/21/11-maneras-de-reutilizar-tu-viejo-telefono-android/

- En 2011 se constata que 8 de cada 10 personas cuenta con un móvil, lo que implica que la venta a través de Internet es accesible por gran parte de la población mundial desde cualquier parte y en cualquier momento.
- Un informe de RetailMeNot[9] de 2014 [32], asegura que en Europa se prevé un aumento de la venta por *e-commerce* del 18,4% para 2015 y en España un 18,6% (ver tabla 1).

[9] RetailMeNot - www.retailmenot.com

Forecast Sales Growth Online and Offline, 2015 (at current prices)			
	Online sales growth	Store-based Sales Growth	All retail sales grow (online + store based)
UK	16.2%	-2.0%	3.5%
Germany	23.1%	-1.8%	2.0%
France	17.0%	-1.7%	1.2%
Spain	18.6%	-0.6%	1.2%
Italy	19.0%	-1.0%	0.3%
Netherlands	16.8%	-1.4%	1.0%
Sweden	15.5%	-1.2%	2.9%
Poland	21.0%	-0.6%	3.6%
Europe	**18.4%**	**-1.4%**	**2.0%**
US	13.8%	-1.9%	3.6%
Canada	13.2%	-0.4%	2.4%

Tabla 1. Previsión de crecimiento de venta online *para los principales países en el año 2015. Fuente: http://www.digitalstrategyconsulting.com/intelligence/2015/01/global_ecommerce_trends_2015_uk_leads_the_way_in_europe_and_north_america.php*

Todos estos datos dan a entender que el *e-commerce* se encuentra actualmente en plena fase de expansión y crecimiento, y es por ello que las compañías invierten gran parte de sus beneficios en implantar y mantener este tipo de sistemas, ya no sólo para crear negocio utilizando nuevos canales, sino como propia herramienta de marketing mostrando sus productos al mundo entero.

Como también se puede ver en la tabla 1, el negocio basado en tienda física retrocede en todos los países, aunque no es directamente achacable al auge y constante crecimiento de la venta *online*, debido a que pueden estar interviniendo otros muchos factores. Ciertamente, muchos vendedores han optado por incorporar el *e-commerce* como valor añadido para sus clientes.

2.2. Productos *E-commerce* existentes

Existen múltiples opciones a la hora de escoger un producto *e-commerce*, en versiones de pago, de código abierto, de pago por suscripción, como extensión de otro producto, etc. También los podemos diferenciar por la tecnología en la que se basan, siendo las más utilizadas PHP y .NET como lenguajes de programación del *frontend* y MySQL y Microsoft SQL Server como base de datos *backend*.

Los principales productos existentes, categorizados por el tipo de licencia son los que se enumeran a continuación.

2.2.1. Licencia de pago

Los productos más populares cuya licencia es de pago, es decir, es necesario comprar los derechos de uso del software, debiéndose instalar en un servidor que proporcione el comprador de la licencia, son:

- **Magento Enterprise Edition**: aunque cuenta con una versión "open source", su versión de pago está dirigida a compañías de venta *online* más grandes. Su poder de integración es quizás su punto fuerte, así como el acceso a soporte experto en la solución, formación y servicios de consultoría. [37]
- **Avactis Professional o Premium[10]:** el lenguaje de programación en que está creado es PHP, y la base de datos es MySQL.

2.2.2. Licencia de pago por uso o *SaaS*

En este grupo están los productos de venta *online* cuyo uso requiere de un pago recurrente, funcionando de manera similar a las suscripciones. En esta

[10] www.avactis.com

modalidad la infraestructura técnica es normalmente proporcionada por el proveedor del servicio.

Podremos encontrar, como productos más populares [22]:

- **Shopify**[11]: en su web aseguran (Julio 2015) que más de 15.000 vendedores *online* utilizan Shopify (ver figura 4), lo que la convierte en uno de los principales productos *e-commerce* de modelo *SaaS*.

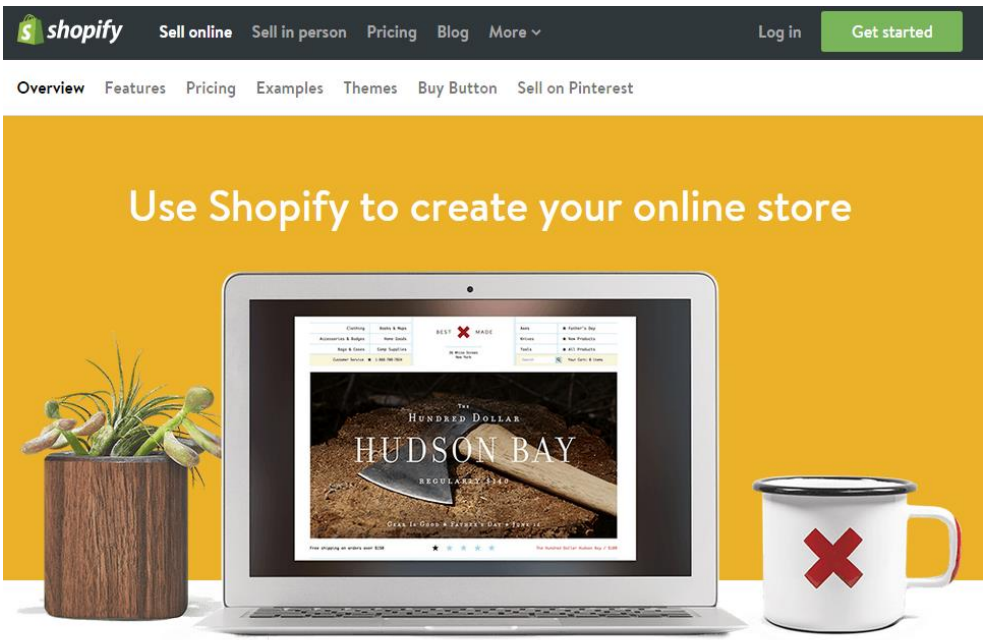

Figura 4. Web de Shopify. Fuente: https://www.shopify.com/

- **Volusion**[12]**:** según su web (Julio 2015) la utilizan más de 30.000 vendedores. A este *e-commerce* se le conoce principalmente por ser fácil de usar y tener una gran fuente de información dirigida a ayudar al usuario, así como soporte 24x7.
- **Demandware**[13]: esta solución *e-commerce* está destinada a vendedores de tamaño medio o grande. Su gran valor está en su alta disponibilidad y

[11] www.shopify.com
[12] www.volusion.com
[13] www.demandware.com

constante desarrollo de actualizaciones que el proveedor se encargará de aplicar de manera transparente para el suscriptor y sus clientes.

- **Venda[14]:** también está dirigido al gran vendedor de manera que pueda reducir su infraestructura tecnológica de manera significativa y poder ofrecer un sistema de venta *online* de alta disponibilidad y mantenimiento integral.
- **NetSuite**[15]: la peculiaridad de este producto es que, aparte de *e-commerce*, ofrece también una solución ERP y CRM. Está dirigido al mediano comercio que pretende acelerar su crecimiento, y cuenta con potentes herramientas de medición de impacto, interacción y tráfico.
- **Big Cartel[16]:** está destinado a ofrecer venta *online* a pequeños artistas, construido para promocionar y vender trabajos artísticos, desde joyas hasta cuadros o ropa.
- **Goodsie[17]:** en este caso hablamos de un *e-commerce* extremadamente sencillo pero muy fácil de poner en marcha sin necesidad de contar con ningún conocimiento técnico previo.

2.2.3. Open source

Los productos de software de tipo "open source" o código abierto, se caracterizan por hacer disponible a la comunidad de desarrolladores el código de la aplicación, para su modificación, ya sea para proyectos independientes, o para mejorar el propio producto.

Dentro de este tipo de productos, y para la categoría *e-commerce* que nos ocupa, tenemos los siguientes productos destacados:

- **Magento Community Edition[18]:** se distribuye gratuitamente como producto con licencia OSL 3.0, en soporte digital mediante descarga de un

[14] www.venda.com
[15] www.netsuite.com
[16] www.bigcartel.com
[17] www.goodsie.com
[18] www.magento.com

fichero ".zip" desde su web (ver figura 5), y está desarrollado sobre tecnología PHP y MySQL. Permite gran versatilidad y está dirigido a desarrolladores y pequeños comercios. Soporta la instalación de extensiones, que pueden incorporarse a un directorio público llamado "Magento Connect[19]".

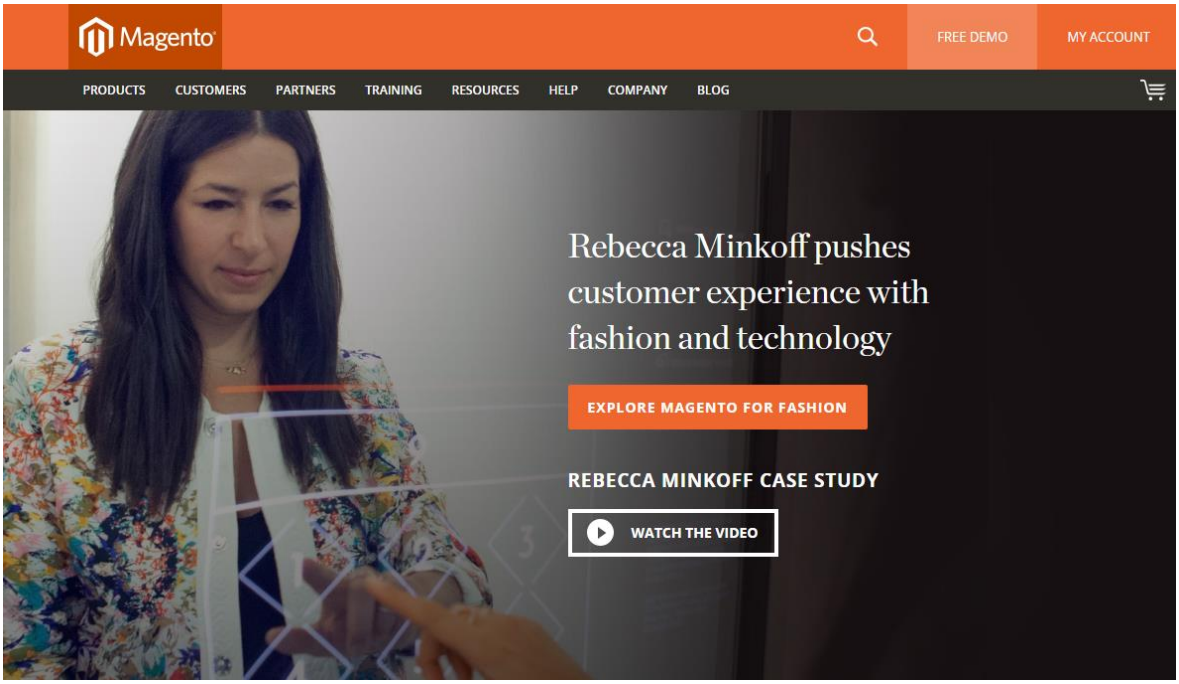

Figura 5. Web de Magento. Fuente: www.magento.com

- **AspxCommerce[20]:** construido sobre la tecnología .Net de Microsoft, el punto fuerte de este *e-commerce* son los sistemas de medición de desempeño de las ventas tan potentes con que cuenta.
- **Prestashop[21]:** al igual que Magento, está construido sobre PHP y MySQL. Este *e-commerce* ha cobrado gran fama desde 2007, año de su lanzamiento, y ya es la tecnología escogida para vender en más de 230.000 tiendas *online*. Destacan su gran fiabilidad y flexibilidad. Admite, a través de un sistema de módulos, extender la funcionalidad con

[19] www.magentocommerce.com/magento-connect/
[20] www.aspxcommerce.com
[21] www.prestashop.com

desarrollos particulares que se pueden buscar en el directorio llamado "Addons Marketplace". Algunos son gratuitos, pero la mayoría son de pago.

- **NopCommerce**: es el software de *e-commerce* en el que se basa la funcionalidad descrita en la presente memoria, por lo que se describe más detalladamente en la siguiente sección.

2.3. NopCommerce

NopCommerce (ver figura 6) es un software open-source del tipo *e-commerce*, que cuenta con un *frontend* y un *backend* de administración. Es completamente personalizable, estable, altamente usable, y cuenta con una extensa documentación que ofrece una amplia base de información, recursos y soporte para la comunidad desarrolladora de nopCommerce.

Figura 6. Sitio web de nopCommerce.

Está basado en .Net MVC y utiliza Microsoft SQL Server 2008 (y versiones posteriores) como base de datos *backend*. Ha sido descargado más de 1,5 millones de veces. Su descarga es gratuita, si bien cuenta con un sitio de extensiones y temas, desarrollados por miembros de la comunidad o el propio equipo de nopCommerce, y que en algunos casos tienen un coste.

Permite la venta de productos de tipo físico y también de tipo digital, la gestión de usuarios tipo B2C y también la aplicación de tarifas B2B.

A continuación se detallan sus principales características, las cuales hacen que se decida por este producto para realizar el presente proyecto.

2.3.1. Soporte para extensiones (*plugins*)

NopCommerce cuenta con un sistema de instalación de extensiones que permite hacer más funcional el producto, y que en este caso ha permitido obtener un beneficio del producto base sobre el cual construir la funcionalidad objeto del proyecto.

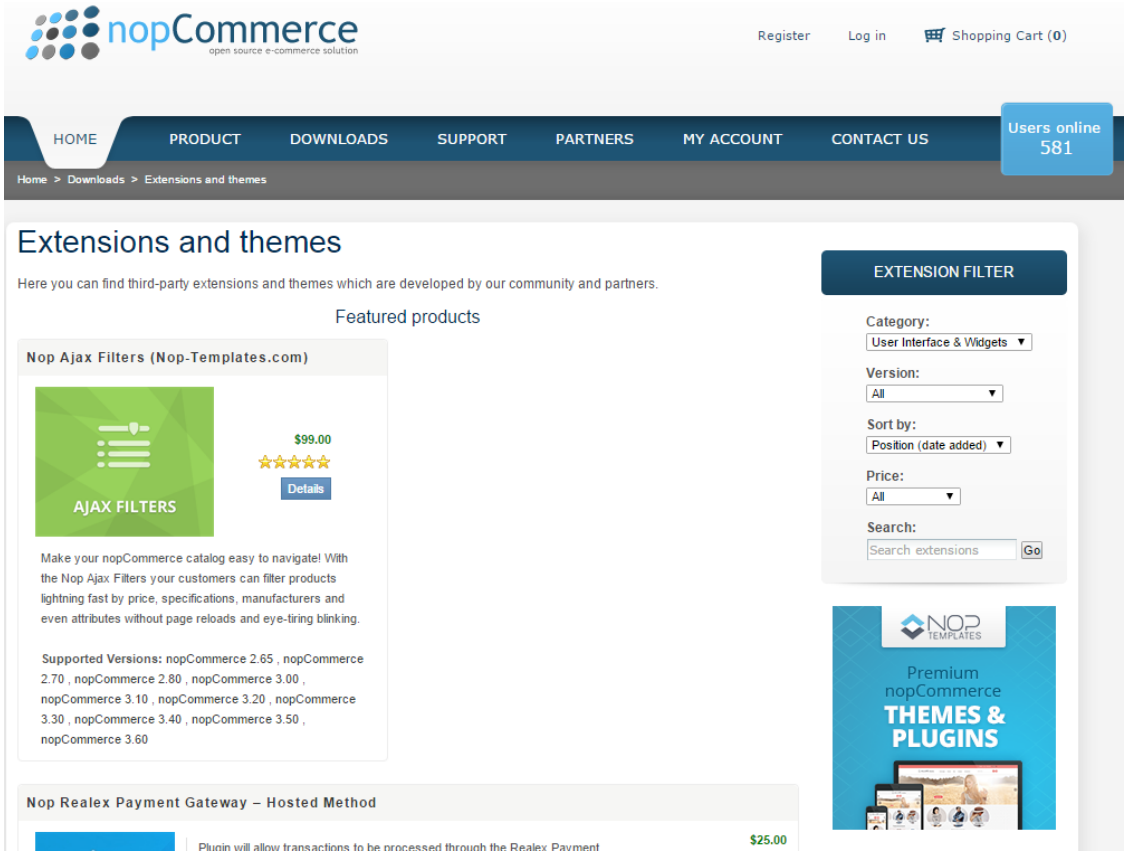

Figura 7. Directorio de extensiones.

En el directorio de extensiones de nopCommerce (ver figura 7) se encuentran extensiones para multitud de aplicaciones, como mejoras en la interfaz de usuario, sistemas de pago con tarjetas de crédito para proveedores específicos, visualizadores de modelados 3D de productos, un slider con testimonios de clientes…

Es importante destacar que la instalación de las extensiones es extremadamente fácil, ya que tiene un sistema de administración específico para ello con un botón con el que instalar y desinstalar. Sólo hay que ubicar el directorio con el contenido de la extensión en una ruta concreta de la aplicación, para que aparezca en el listado de extensiones disponibles del sistema de administración mencionado.

Hoy en día, y especialmente con productos de tipo open source, se hace indispensable la posibilidad de ampliar la funcionalidad del producto base de manera que otros desarrolladores, en un modelo de parternship, puedan beneficiarse de crear estas extensiones y venderlas desde el directorio de nopCommerce en su web oficial.

2.3.2. Gestión de productos

NopCommerce permite crear nuevas características de producto sin tener que modificar la base de datos, mediante atributos genéricos, y de los que se sirve la implementación para publicar las propiedades de los libros obtenidas de Google Books API. También cuenta con funcionalidades interesantes relacionadas con el producto, como:

- Productos descargables como puede ser música o libros electrónicos mediante el adjuntado de licencias o acuerdos de usuario, lo que podría ser parte de un proyecto futuro utilizando el nuestro como base
- Categorías y fabricantes que facilitan la organización de los productos dentro del catálogo de nopCommerce. La existencia de una interfaz de manejo de estas categorías nos ha permitido poder gestionar automáticamente la asignación de cada uno de los libros a sus categorías BISAC que Google Books tiene informadas.

- Productos de tipo alquiler. Podría ser parte igualmente de un proyecto futuro, la gestión del alquiler de los libros provenientes de Google Books para, por ejemplo, un sistema de gestión de bibliotecas.
- Configuración de paquetes de productos, como puede ser un equipo informático o packs. Google Books API
- Productos recurrentes para facilitar a los usuarios que compran frecuentemente un producto el tenerlo más accesible
- Etiquetas para poder encontrar rápidamente productos del mismo tipo pinchando en dicha etiqueta que deberá ser lo más descriptiva posible
- RMA's, para la gestión de devoluciones de producto
- Asociar varias imágenes al mismo producto, característica muy útil principalmente en productos que requieren de una mejor descripción gráfica, como bicicletas, instrumentos musicales…
- Incorpora widgets para mostrar productos de una manera más destacada, y marcar los que están en oferta o son nuevos
- Deshabilitar el botón de comprar para ciertos productos
- Especificaciones de producto para aportar mayor información al usuario, como en los casos de productos tecnológicos en los que es importante dar detalles de sus características
- Ventas cruzadas del tipo "otros usuarios también compraron" y "productos relacionados"
- ACL para productos, categorías y fabricantes
- Compartir por correo electrónico
- Dependencia entre productos a la hora de agregar al carrito, de manera que sea necesario que otros productos sean igualmente agregados
- Divisa con cambio en tiempo real
- Medias, pesos y dimensiones de producto configurables
- Importar y exportar productos
- Edición de productos en modo masivo

2.3.3. Construido sobre tecnología ASP.Net y MVC

Otra de las características que contribuye a tomar una decisión por nopCommerce, es que está desarrollado en tecnología ASP.Net y basado en el patrón de desarrollo MVC, lo que proporciona los siguientes beneficios:

- Escalabilidad debido a la posibilidad de separar completamente presentación de lógica de negocio
- Existencia de un *Framework* para Google Books API en ASP.Net, permitiendo así la posibilidad de integrar la API directamente en el desarrollo de la extensión. De otro modo habría sido necesario el uso de su REST API a través de llamadas HTTP, pero el empleo del *Framework* hace la aplicación más robusta y el desarrollo más sencillo.
- Conocimiento de la tecnología por parte del autor del proyecto. En mi carrera laboral he utilizado a diario ASP.Net y basado en MVC más últimamente.

2.3.4. Otras características

Las características de nopCommerce son muy numerosas, y las que se han enunciado son las principales que nos han ayudado en el proyecto y las que han hecho que nos decantásemos por este producto, pero hay más funcionalidades, también muy interesantes y que hacen de este producto una buena elección a la hora de escoger un *e-commerce* open source. Entre estas funcionalidades adicionales, destacan:

- Proceso de compra en un clic.
- Herramientas para gestionar el SEO (Search Engine Optimization).
- Marketing: herramientas para obtener información acerca del impacto de nuestros productos en el mercado, para hacer campañas de marketing, etc.
- Integración con más de 50 sistemas de pago *online*, agregables mediante extensiones.

- Cobro demorado, que permite almacenar la información de la transacción con tarjeta de crédito y no ejecutarla hasta que el producto ha sido entregado.
- Integración con los sistemas de reparto más populares, como UPS[22] y FedEx[23].
- Gestión de múltiples impuestos, por producto, tipo de producto, cliente, país, provincia y código postal, e integración con sistemas *online* de cálculo de impuestos, útiles en Estados Unidos y Canadá.
- En el aspecto de atención al cliente, incorpora infinidad de características para mejorar la comunicación entre vendedor y cliente, o revendedor/distribuidor. Entre ellas están las wishlist o lista de favoritos, soporte para lectura derecha-izquierda, soporte para la ley de cookies europea, web services para permitir a revendedores o distribuidores acceder a información del *backend*, chat en vivo, zonas horarias, recuperación de contraseña, log de actividad de cliente…

[22] www.ups.com
[23] www.fedex.com/es

2.4. Google Books

Desde los inicios de Google, fundada por Sergey Brin y Larry Page, en 1996, la idea de Google Books estuvo siempre presente. Trabajando para un proyecto financiado por Standford Digital Library Technologies Project[24], su objetivo era crear bibliotecas digitales de la siguiente manera: pensando en que en el futuro el número de libros digitales sería inmenso e iría creciendo, mediante un rastreador se podrían indizar los libros digitales existentes analizando las conexiones entre ellos, definiendo la relevancia y la utilidad del contenido por el número de veces y calidad de las citas encontradas sobre un libro a indizar desde otros libros.

Crearon este rastreador, al que llamaron Backrub (ver figura 8), y esta tecnología fue la que inspiró lo que hoy es el motor de búsqueda de Google como lo conocemos.

Figura 8. Aspecto de Backrub. Fuente: http://www.informationin.com/2014/10/google-was-once-named-backrub.html

La Universidad de Standford lo estuvo utilizando durante un año, pero finalmente lo dio de baja debido al ancho de banda que consumía.

Por aquel entonces, Larry y Sergey ya veían en sus mentes a gente de todas partes buscando libros digitales de todo el mundo para encontrar aquellos que les interesaban.

[24] http://diglib.stanford.edu:8091/

Siguiendo con la historia de Google Books, en 2002 un grupo de usuarios de Google creó el proyecto secreto "Books", y hablando con expertos sobre los desafíos a los que se enfrentaban el asunto central a tratar era cuánto tiempo llevaría digitalizar todos los libros existentes. Obviamente, no había una respuesta posible. Por lo que Larry Page decidió investigar por sí mismo, y utilizó un método sencillo con el que hacer un cálculo básico. Junto a Marissa Mayer, una de los primeros product managers de Google, y utilizando un metrónomo para poder mantener un ritmo, contaron las páginas de un libro de 300 páginas. Les llevó 40 minutos. Visitaron varios proyectos en aquel momento existentes, como el **American Memory de la Biblioteca del Congreso**[25]**, Proyecto Gutenberg**[26] (en memoria del célebre inventor de la imprenta en 1437) y que ahora dispone de más de 49.000 libros electrónicos gratuitos, o **El Proyecto del Millón de Libros**[27] sobre digitalización de libros para ver la manera en que trabajaban.

Como parte de esta búsqueda e investigación, Larry llega a la **Universidad de Michigan**[28], pionera en proyectos de digitalización de bibliotecas, como **JSTOR**[29] o **Making of America**[30]**.** Cuando llega a la conclusión de que digitalizar la biblioteca de la Universidad, de 7 millones de libros, llevaría 1.000 años, le ofrece a la presidenta, Mary Sue Coleman, el buscador Google para facilitar la tarea y que sea posible en tan solo 6 años.

Ya en 2003, un miembro de su equipo se desplaza a una feria de libros en Phoenix, Arizona, para adquirir libros que le sirvieran para testear técnicas de escaneo no destructivas. Después de incontables intentos, el equipo desarrolla un método de escaneo mucho más respetuoso con el libro que el actual proceso de alta velocidad.

[25] http://memory.loc.gov/ammem/index.html
[26] http://www.gutenberg.org/
[27] https://archive.org/details/millionbooks
[28] https://www.umich.edu/
[29] http://www.jstor.org/ - https://twitter.com/JSTOR
[30] http://quod.lib.umich.edu/m/moagrp/

Al mismo tiempo, hacen progresos para resolver los problemas técnicos derivados de libros con impresiones incorrectas o caracteres poco claros, además de cualquier otra incidencia aparecida, en 430 idiomas distintos.

Al año siguiente, en 2004, visitan una librería que data del año 1602 en que Sir Thomas Bodley la fundó, y quedaron tan impresionados con las obras que allí almacenaban que les dio un nuevo impulso para poder poner al alcance de cualquier persona aquellas obras tan antiguas y que raramente habían visto la luz del día. Llegaron a un acuerdo con la librería para digitalizar la colección de más de un millón de libros del siglo XIX en los siguientes 3 años.

Mientras tanto, llegan a acuerdos también con algunas de las más importantes editoras y en Octubre, Larry y Sergey, anuncian la creación de "Google Print", en la **Feria del Libro de Frankfurt**[31], en Alemania. Se unen al programa entidades tan importantes como McGraw-Hill o Blackwell. Ese mismo Diciembre, Google anuncia el inicio del **Print Library Project**[32], posible gracias a los acuerdos establecidos con Harvard, Universidad de Michigan, la biblioteca pública de Nueva York, Oxford y Standford.

En 2005, de nuevo en la Feria del Libro de Frankfurt anuncian los acuerdos a los que están llegando con partners de ochos países europeos, para atraer un mayor número de colaboraciones.

Este mismo año, Google dona 3 millones de dólares a la Biblioteca del Congreso de Estados Unidos para ayudar a la creación de la Biblioteca Mundial Digital, que facilitará acceso *online* a colecciones raras y únicas desde cualquier parte del mundo. Extienden el proyecto piloto iniciado previamente con esta biblioteca para digitalizar también libros de valor histórico de la Biblioteca Legal del Congreso.

También en 2005, Google Print cambia su nombre al nombre actual Google Books, que representa de manera más acertada cómo los usuarios utilizan el servicio.

[31] http://www.book-fair.com/en/
[32] http://books.google.com/googlebooks/library/index.html

Poco más tarde, Google empezaba a plantear con algunos *partners* la idea de dar acceso bajo suscripción a todos los libros a través del navegador, para así ayudar a las editoras a experimentar con nuevas maneras de vender libros *online*.

Se da acceso a todos los trabajos escaneados que carezcan de *copyright*, agregando a los resultados de la búsqueda de Google Books un botón para descargar el contenido en PDF y se mejora la interfaz del sistema, agregando una página "Acerca de este libro" (ver figura 9) que publica páginas con datos de cada libro en un formato amigable, utilizando el algoritmo de Google.

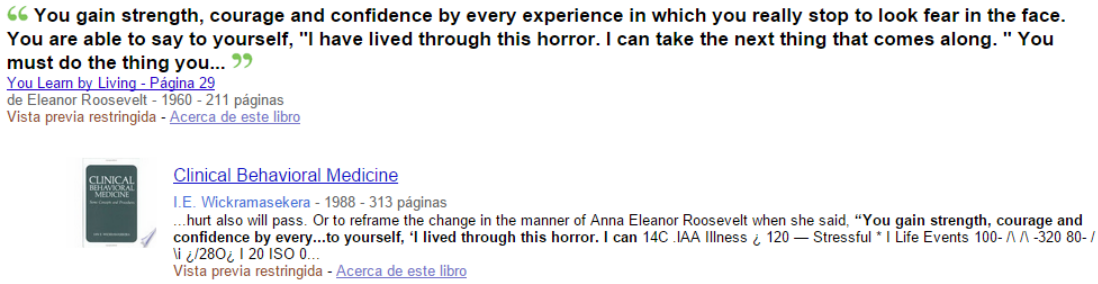

Figura 9. Desde dónde acceder a "Acerca de este libro". Fuente: Google.

Ese otoño cuatro nuevas bibliotecas de Universidades se adhieren al programa: Universidad de California, Universidad Complutense de Madrid, Universidad de Wisconsin y la Universidad de Virginia.

Desde entonces y hasta ahora se han ido desarrollando funcionalidades muy interesantes de las cuales algunas persisten y otras han desaparecido, aunque la evolución del producto parece que sufrió una desaceleración. Otros aspectos importantes de Google Books ocurridos desde entonces, es que se encuentra integrado con la tienda de Google "**Google Play**[33]", que se han creado nuevos acuerdos de manera constante para aumentar la base de datos de libros existente, o

[33] https://play.google.com/store/books

librerías personalizadas para el usuario Google, llamada "Mi Librería", donde marcar los libros que a dicho usuario le parezcan más interesantes.

Las funcionalidades más significativas son las que se integran en este proyecto: Google Books API y Embedded Viewer API, que explicamos brevemente a continuación. [23] [24]

2.4.1. Google Books API

Como su propio nombre indica, Google Books cuenta con una API para facilitar el acceso por terceros a la información del directorio de libros de Google.

Una API o interfaz de programación de aplicaciones, nos posibilita llamar a procedimientos y funciones definidos en un software para crear una abstracción de lo que dentro de dicho software se ejecuta publicando una interfaz de qué operaciones pueden ejecutarse sobre el sistema.

Las APIs son un recurso muy útil para interconectar programas y servicios y así beneficiarse, por ejemplo un programa A, de la funcionalidad que implementan sin tener que volver a programarlo en dicho programa A, ya que el sistema que implementa la API nos va a proporcionar las salidas que precisamos.

La interfaz únicamente publica el nombre de cada una de las funciones que la forman, así como los parámetros de entrada y el tipo de datos de salida.

En el caso de Google Books API, proporciona la mayoría de operaciones con que cuenta el sitio de Google Books, pero las principales son:

- buscar y navegar por la lista de libros que coinciden con una búsqueda dada.
- ver información sobre un libro, incluyendo metadatos, disponibilidad y precio, y enlaces a la página de pre visualización.
- administrar tus propios "bookshelves" que son directorios personales de los libros que el usuario marca.

Las peticiones a Google Books API privadas, como la gestión del "bookshelf" o la crítica de un libro, deben ser autenticadas, es decir, que el sistema cliente debe utilizar el proceso de autenticación OAuth 2.0 que se lanzará en el momento de iniciar una petición a la API, para que el usuario autorice, estando "logado" en su cuenta de Google, la utilización de dicha cuenta por parte de la API.

El proceso de autenticación por **OAuth**[34] 2.0, sigue unas pautas que no siempre tienen que ser las mismas, pero en modo general son:

1. Al crear la aplicación que va a consumir la API, el desarrollador debe registrarla en la **Google Developer Console**[35], obteniendo un ID de cliente y "client secret" que es un token que se utilizará en las comunicaciones de autenticación.
2. Activar la API de Google Books en la Google Developer Console.
3. Cuando la aplicación precisa acceso a datos de usuario (privados), pregunta, preguntará a Google por el ámbito en el que se utilizará el acceso.
4. Google presentará una pantalla de consentimiento al usuario, preguntándole por autorización de acceso de la aplicación a los datos de usuario que precisa.
5. Si el usuario lo autoriza, Google proporciona a la aplicación un token de acceso de corta duración.
6. La aplicación entonces solicitará datos de usuario adjuntando el token de acceso a la petición.
7. Si Google determina que la petición y el token son válidos, contestará con los datos solicitados.

Las operaciones que en la web de Google Books se realizan de manera pública, como pueden ser búsquedas generales o consulta de un libro, no requieren de autenticación, pero sí de un "API key" que se vincula a una cuenta de Google, y que es utilizada para controlar el volumen de peticiones que se lanzan contra la API.

[34] http://oauth.net/ - https://www.googleapis.com/auth/books
[35] https://console.developers.google.com/

Esta clave API debe acompañarse con las peticiones utilizando un parámetro llamado "key" de manera que si se realiza una petición por URL, se debe agregar una cadena del tipo "`key=miAPIkey`" sin codificación alguna.

Hay tres tipos de IDs utilizados en el API:

- Volume IDs: cada volumen del directorio de Google Books tiene un identificador único.
- Bookshelf IDs: son enteros con los que se identifica cada uno de los tipos de Bookshelf o librería que tiene Google Books:
    - Favoritos: 0
    - Comprados: 1
    - Por leer: 2
    - Leyendo: 3
    - Leídos: 4
    - Reseñados: 5
    - Recientemente vistos: 6
    - Mis eBooks: 7
    - Libros recomendados: 8

  El sistema soporta la creación de Bookshelves customizados, que tendrán identificador mayor que 1000.

- User IDs: cada usuario de Google tiene un identificador único para Google Books, que actualmente únicamente se puede obtener del campo "selfLink" de un recurso "Bookshelf" obtenido con una petición autenticada.

2.4.2. Alternativas

Se han evaluado otras alternativas antes de seleccionar Google Books, como fueron:

- **Biblioeteca**[36]: es un portal de recomendación de libros (ver figura 10), con una capa social, basado en España. Ofrece también una API de acceso a su funcionalidad, pero se estimó en su momento que la información que retornaba no era suficiente para hacer un proyecto consistente. No era posible obtener, por ejemplo, el número de páginas de un libro o la editorial, que en muchos casos no aparecía o había que buscarla en el campo de descripción larga. Debido a esto fue descartada.

Figura 10. Sitio web de Biblioeteca.

- **Amazon API**[37]: la API del sitio de Amazon está orientado a la obtención de información para publicación de anuncios en otros sistemas, si bien, tras una breve investigación, resultaba ser de uso, el manual de la API en mi opinión no era nada claro y no se disponían de ejemplos en la tecnología escogida .Net MVC. Por estas dos razones, principalmente, fue también descartada.

[36] www.biblioeteca.com
[37] https://developer.amazon.com/public/apis

2.5. BISAC *Subject Headings*

BISAC (*Book Industry Standards and Communications*) *subject headings* se trata de un estándar de categorización de obras escritas en formato físico y digital.

Cuenta con un listado de materias aprobadas por organismos de la industria del libro, representados por un identificador alfanumérico de nueve caracteres. Cada materia puede representar uno o varios niveles de descripción, habiendo 52 niveles principales, tales como "*COMPUTERS*", "*FICTION*", "*HISTORY*", "*EDUCATION*"... Haciendo una representación jerárquica podríamos incluir otras materias dentro de estos niveles principales pero con un nivel adicional de detalle. Por ejemplo *"ART / History / General*". [12]

La idea inicial era la de estandarizar las materias de las obras cuando la información relativas a estas obras es transferida electrónicamente. Actualmente, el sistema de Google Books utiliza este estándar para categorizar sus libros, asignando a cada uno de ellos de 1 a n materias en las que puede estar incluida la obra. Otras entidades que utilizan este estándar son Amazon, **Baker & Taylor**[38], **Barnes & Noble**[39], **Nielsen Bookscan**[40], **Bowker**[41], **Indiebound**[42], **Indigo**[43], **Ingram** [44]y la mayoría de los grandes editores. [31]

El listado está mantenido por BISG (Book Industry Study Group), asociación líder en investigación sobre mejores prácticas, obtención de información, educación y eventos sobre libros. Facilita conexiones para resolver problemas comunes, avanzar nuevas ideas, y ofrecer nuevo contenido publicado a los lectores. Abarca al consumidor, educador, profesional y divulgadores productores de contenidos en

[38] www.bayker-taylor.com
[39] www.barnesandnoble.com
[40] www.nielsenbookscan.co.uk
[41] www.bowker.com
[42] www.indiebound.org
[43] https://book.goindigo.in
[44] www.ingramcontent.com

formato digital o físico. Incluye a más de 200 compañías desde 5 grandes hasta compañías de tecnología *start-ups*. Fundada en 1975 en la conferencia anual del *Book Manufacturers' Institute*[45] (Florida, Estados Unidos), para mejorar la capacidad investigadora, continúa evolucionando publicando, por ejemplo, un listado de materias actualizado de periodicidad anual.

Gestiona también otros estándares, como *ONIX for Books* o el estándar X12 de transacciones para *e-commerce* y publica recomendaciones y mejores prácticas para el uso de identificadores como **ISBN**[46], EAN, etc.

En el presente proyecto el listado de BISAC se utiliza para ofrecer al usuario administrador la posibilidad de elegir qué categorías utilizar para generar la librería *online*. De esta manera se puede personalizar las materias con las que el nuevo vendedor quiere salir al mercado.

[45] http://www.bmibook.org
[46] https://www.isbn-international.org/content/isbn-users-manual

2.6. SignalR

SignalR [47] es una librería para la tecnología ASP.Net que posibilita el envío de peticiones desde el servidor web al cliente y viceversa, lo que permite crear aplicaciones que se ejecutan y presentan resultados en tiempo real.

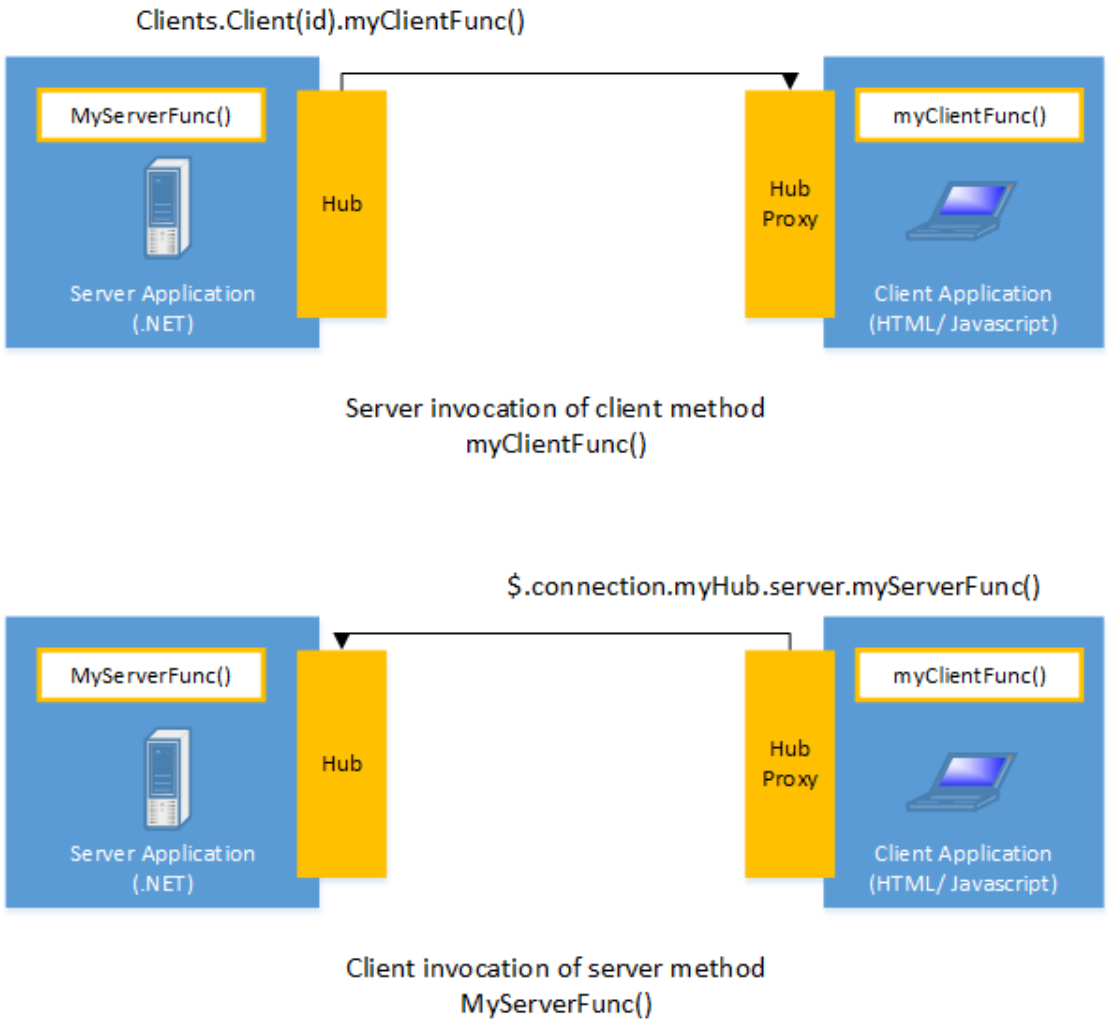

Figura 11. Esquema de comunicación de SignalR. Fuente: www.asp.net

Se utiliza generalmente para informar al usuario del proceso que el servidor está llevando a cabo tras haber recibido una petición desde el cliente, pero realmente

[47] http://signalr.net/

implementa el protocolo RPC [25], por lo que el cliente también cuenta con procedimientos que el servidor puede ejecutar y tiene, por tanto, infinidad de aplicaciones: actualización simultánea de un campo en todos los navegadores sin necesidad de que estos refresquen la página, actualizar el contenido mostrado de un *feed* de datos remoto cuando este cambia sin necesidad de refrescar la página...

En un momento determinado se abre una conexión persistente entre el cliente y el "concentrador" o "*hub*" en el servidor, y "*hub proxy*" en el cliente, que son una abstracción de lo que en tecnología de la información se conoce como "*socket*" y que se encarga de gestionar los mensajes RPC que recibe y envía.

En el presente proyecto se emplea para mantener al usuario informado del proceso de creación de la librería, ya que lleva un tiempo de procesado durante el cual, sin SignalR, el usuario no habría sabido si el servidor continuaba la ejecución o no. En una aplicación web, si el usuario envía un formulario y percibe que la aplicación no le devuelve una contestación, pensará que algo va mal y le generará desconfianza. Por ello se ha implementado esta solución.

3. Análisis

El análisis es, quizás, la fase más importante del ciclo de vida de software, ya que depende de una correcta toma de requisitos el que el resto del proyecto discurra sin problemas como los producidos por una especificación ambigua de lo que debe hacer el sistema. En este capítulo se aborda esta fase del ciclo de vida software.

3.1. Requisitos del sistema

Los requisitos del sistema definen qué es lo que debe hacer el sistema a petición del usuario final, utilizando un lenguaje natural de manera que sirva de entendimiento entre el usuario final y el desarrollador.

Es importante en esta definición de requerimientos evitar cualquier tipo de ambigüedad a la hora de redactar cada una de las funciones, ya que un requisito que puede entenderse de varias maneras es probablemente un problema en el futuro. El usuario final piensa en cómo debe funcionar un sistema de distinta manera a cómo lo hace un desarrollador, y debido a la dificultad que existe en analizar dicho sistema de manera que ambas partes lo entiendan la definición de requisitos es siempre susceptible de ambigüedades.

Debe quedar muy claro qué hará el programa y qué no hará, de manera que cualquier modificación que solicite el usuario final quede bien ubicado fuera de los requerimientos iniciales, principalmente para evitar costes no contemplados en el presupuesto del proyecto y que la parte desarrolladora tuviera que asumir.

3.1.1. Requisitos tecnológicos

Son las necesidades tecnológicas que precisa la extensión y, por tanto, nopCommerce, para poder satisfacer un correcto funcionamiento. Aunque siempre es preferible establecer unos requisitos que permitan al sistema funcionar con holgura,

hay que tener en cuenta el coste de los equipos que son más costosos a medida que son más potentes, habrá, por tanto, que ajustarlos de manera equilibrada.

Por parte del usuario, la única herramienta necesaria para poder utilizar la extensión es un navegador web moderno.

Entorno usuario	
Navegador web	Microsoft Internet Explorer 8 o superior Mozilla Firefox 2.0 o superior Google Chrome 1.x Apple Safari 2.x

Tabla 2. Entorno de usuario.

En el lado servidor estas son las especificaciones que soporta la aplicación web necesarias para poder desarrollar y ejecutar nopCommerce.

Entorno desarrollador	
Equipo Servidor	● 8 GB de RAM ● Almacenamiento de 100 GB ● Procesador AMD FX - 4170 Quad-Core a 4.20 GHz ● Tarjeta de red 100/10 Mb
Sistema Operativo	● Windows 8 ● Windows 7 ● Windows Server 2008 o 2008 R2 ● Windows Server 2012 o 2012 R2
Frameworks	● ASP.NET 4.5 con MVC 5.0 ● Microsoft .NET Framework 4.5.1. o superior
Motor de base de datos	● Microsoft SQL Server 2008 o superior ● Microsoft SQL Server Compact 4.0 o superior
Software	● NopCommerce 3.5
Servicios web	● Google Books API
Herramientas de desarrollo (IDE)	● Visual Studio 2012 o superior

Tabla 3. Entorno desarrollador.

3.1.2. Requisitos de usuario

En esta sección se detallan los requerimientos que plantea el usuario, cómo espera que funcione el sistema, las funcionalidades que debe cubrir el desarrollo. El formato de cada uno de los requisitos, es el siguiente modelo:

Tabla 4. Formato de tabulación de requisitos de usuario.

ID: identificador	Nombre del requisito
Descripción	Breve descripción de la petición del usuario final
Necesidad	Grado de necesidad para el usuario final de este requisito
Prioridad	Prioridad dada por el usuario. De utilidad para organizar el orden de las tareas.

El identificador es único, denotado por los caracteres “RU-“, más un numeral de dos dígitos que empieza por “01”, siguiendo un orden ascendente en la parte numérica.

ID: RU-01	Ámbito de la aplicación
Descripción	Toda la funcionalidad especificada debe formar parte de una extensión para nopCommerce
Necesidad	Esencial
Prioridad	Alta

Tabla 5. RU-01. Ámbito de la aplicación.

ID: RU-02	Instalación y desinstalación
Descripción	Se debe poder instalar y desinstalar desde el listado de extensiones
Necesidad	Esencial
Prioridad	Alta

Tabla 6. RU-02. Instalación y desinstalación.

ID: RU-03	Configuración
Descripción	Debe ser posible configurar unos valores mínimos de configuración para la extensión
Necesidad	Esencial
Prioridad	Alta

Tabla 7. RU-03. Configuración.

ID: RU-03	Configuración - Accesibilidad
Descripción	Se debe poder acceder a la configuración desde el listado de extensiones
Necesidad	Esencial
Prioridad	Alta

Tabla 8. RU-03. Configuración - Accesibilidad.

ID: RU-04	Extensión - Accesibilidad
Descripción	Se debe crear un menú independiente para la extensión accesible desde el panel de administrador
Necesidad	Esencial
Prioridad	Alta

Tabla 9. RU-04. Extensión - Accesibilidad.

ID: RU-05	Submenú búsqueda de libros
Descripción	Se debe crear un submenú de búsqueda de libros accesible desde el panel de administrador
Necesidad	Esencial
Prioridad	Media

Tabla 10. RU-05. Submenú búsqueda de libros.

ID: RU-06	Submenú configuración de extensión
Descripción	Se debe crear un submenú de configuración accesible desde el panel de administrador
Necesidad	Esencial
Prioridad	Media

Tabla 11. RU-06. Submenú configuración de extensión.

ID: RU-07	Submenú creación de tienda
Descripción	Se debe crear un submenú que lleve a la página de creación de una tienda, accesible desde el panel de administrador
Necesidad	Esencial
Prioridad	Media

Tabla 12. RU-07. Submenú configuración de extensión.

ID: RU-08	Crear literales como etiquetas
Descripción	Los literales utilizados deben integrarse con las etiquetas de nopCommerce
Necesidad	Esencial
Prioridad	Baja

Tabla 13. RU-08. Crear literales como etiquetas.

ID: RU-09	Búsqueda de libros
Descripción	Se debe poder buscar un libro independiente por cualquier cadena, por ISBN y filtrando por categoría del estándar BISAC
Necesidad	Esencial
Prioridad	Alta

Tabla 14. RU-09. Búsqueda de libros.

ID: RU-10	Buscar en directorio de Google Books
Descripción	La búsqueda debe ejecutarse contra el directorio de libros de Google Books
Necesidad	Esencial
Prioridad	Alta

Tabla 15. RU-10.Buscar en directorio de Google Books.

ID: RU-11	Paginar resultados
Descripción	El resultado de la búsqueda debe paginarse
Necesidad	Opcional
Prioridad	Baja

Tabla 16. RU-11. Paginar resultados.

ID: RU-12	Contenido del resultado
Descripción	En el resultado de la búsqueda deben poder verse la imagen de portada, el título, el autor, la descripción de la obra, el ID de Google Books
Necesidad	Esencial
Prioridad	Media

Tabla 17. RU-12. Contenido del resultado.

ID: RU-13	Búsqueda - Peticiones a servidor con AJAX
Descripción	La búsqueda debe realizarse por actualización parcial de la ventana de resultados, en lugar de actualizar la página entera, para dar una mayor sensación de dinamicidad a la aplicación, por ello se deben realizar llamadas al servidor mediante AJAX
Necesidad	Opcional
Prioridad	Media

Tabla 18. RU-13. Búsqueda - Peticiones a servidor con AJAX.

ID: RU-14	Crear categorías en instalación
Descripción	En la instalación debe poder ejecutarse un script SQL que cree las categorías del estándar BISAC, así como las etiquetas multilenguaje
Necesidad	Esencial
Prioridad	Alta

Tabla 19. RU-14. Crear categorías en instalación.

ID: RU-15	Eliminar categorías en instalación
Descripción	En la desinstalación debe poder ejecutarse un *script* SQL que elimine las categorías del estándar BISAC, así como las etiquetas multilenguaje.
Necesidad	Esencial
Prioridad	Alta

Tabla 20. RU-15. Crear categorías en instalación.

ID: RU-16	Atributos de libro editables
Descripción	Los atributos que van a utilizarse del recurso libro obtenido de Google Books API, deben ser modificables por el administrador desde el panel de control en la edición de producto.
Necesidad	Esencial
Prioridad	Alta

Tabla 21. RU-16.Atributos de libro editables.

ID: RU-17	Atributos visibles en ficha de producto
Descripción	Los atributos que van a utilizarse del recurso libro obtenido de Google Books API, deben mostrarse en la ficha de producto del *e-commerce*.
Necesidad	Esencial
Prioridad	Media

Tabla 22. RU-17.Atributos visibles en ficha productos.

ID: RU-18	Botón agregar libro
Descripción	El listado resultado de la búsqueda debe incorporar un botón en la celda derecha para poder agregar individualmente cada libro a la tienda.
Necesidad	Esencial
Prioridad	Alta

Tabla 23. RU-18.Botón agregar libro.

ID: RU-19	Automatizar alta de recursos
Descripción	La acción de agregar libro a la tienda, debe automáticamente asociar las imágenes, crear las categorías que Google Books especifica en caso de que no existan y vincular los atributos del libro
Necesidad	Esencial
Prioridad	Alta

Tabla 24. RU-19.Automatizar alta de recursos.

ID: RU-20	"Crear tienda" informa al usuario
Descripción	Desde "Crear tienda" debe ejecutar la creación de la tienda mediante AJAX e informar al usuario del progreso
Necesidad	Opcional
Prioridad	Media

Tabla 25. RU-20. "Crear tienda" informa al usuario.

ID: RU-21	Al crear una tienda se debe crear el sitio en el servidor web IIS automáticamente
Descripción	Cuando se crea una nueva tienda *online*, automáticamente debe crearse el sitio en el IIS evitando así el trabajo de generarlos manualmente
Necesidad	Opcional
Prioridad	Baja

Tabla 26. RU-21. Al crear una tienda se debe crear el sitio en el servidor web IIS automáticamente.

3.2. Casos de uso

Los casos de uso son una simplificación de quién interactúa con qué en el sistema. Suele ser útil para mostrar al usuario final en esta parte del proyecto, qué funcionalidades van a implementarse y los actores y, de esta manera, tenga una visión general del programa.

Para la elaboración de este diagrama tan utilizado en Ingeniería de Software, la explicación se basa en el estándar UML 2.0.

- Diagrama de casos de uso, nivel 0:

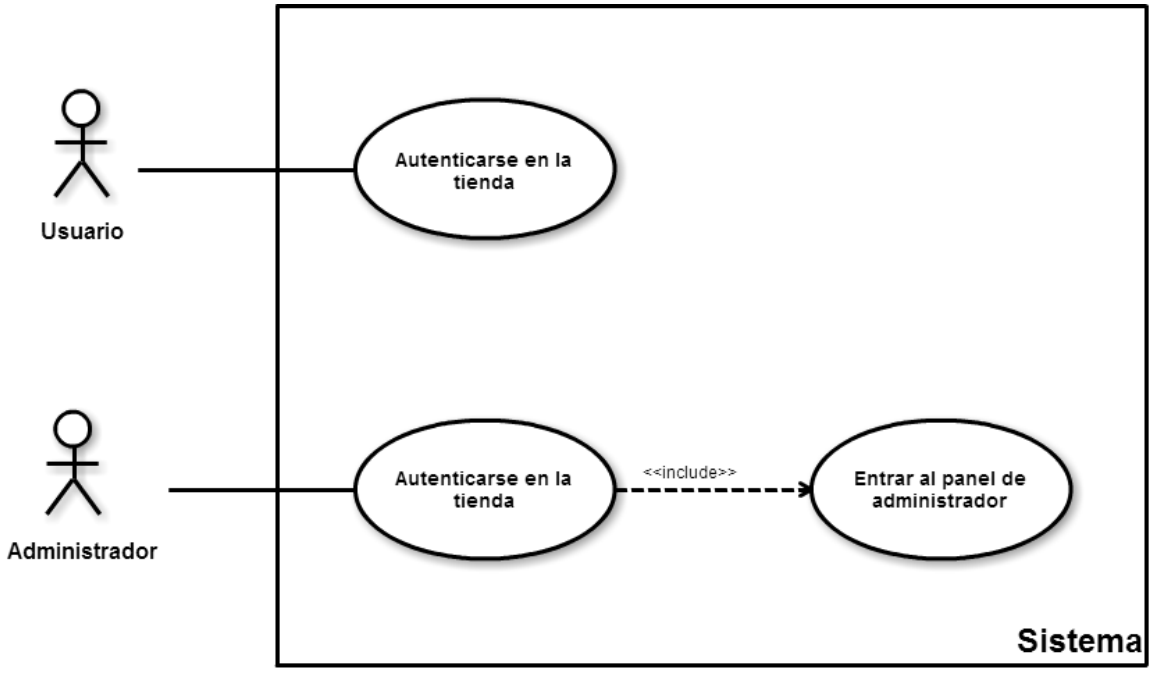

Figura 12. Diagrama de casos de uso. Nivel 0.

- **Autenticarse en la tienda:** proceso por el cual un usuario se identifica en el sistema mediante un usuario y una clave.
- **Entrar al panel de administrador:** dirigirse al área de NopCommerce con las funcionalidades de administración de la aplicación.

- Diagrama de casos de uso, nivel 1:

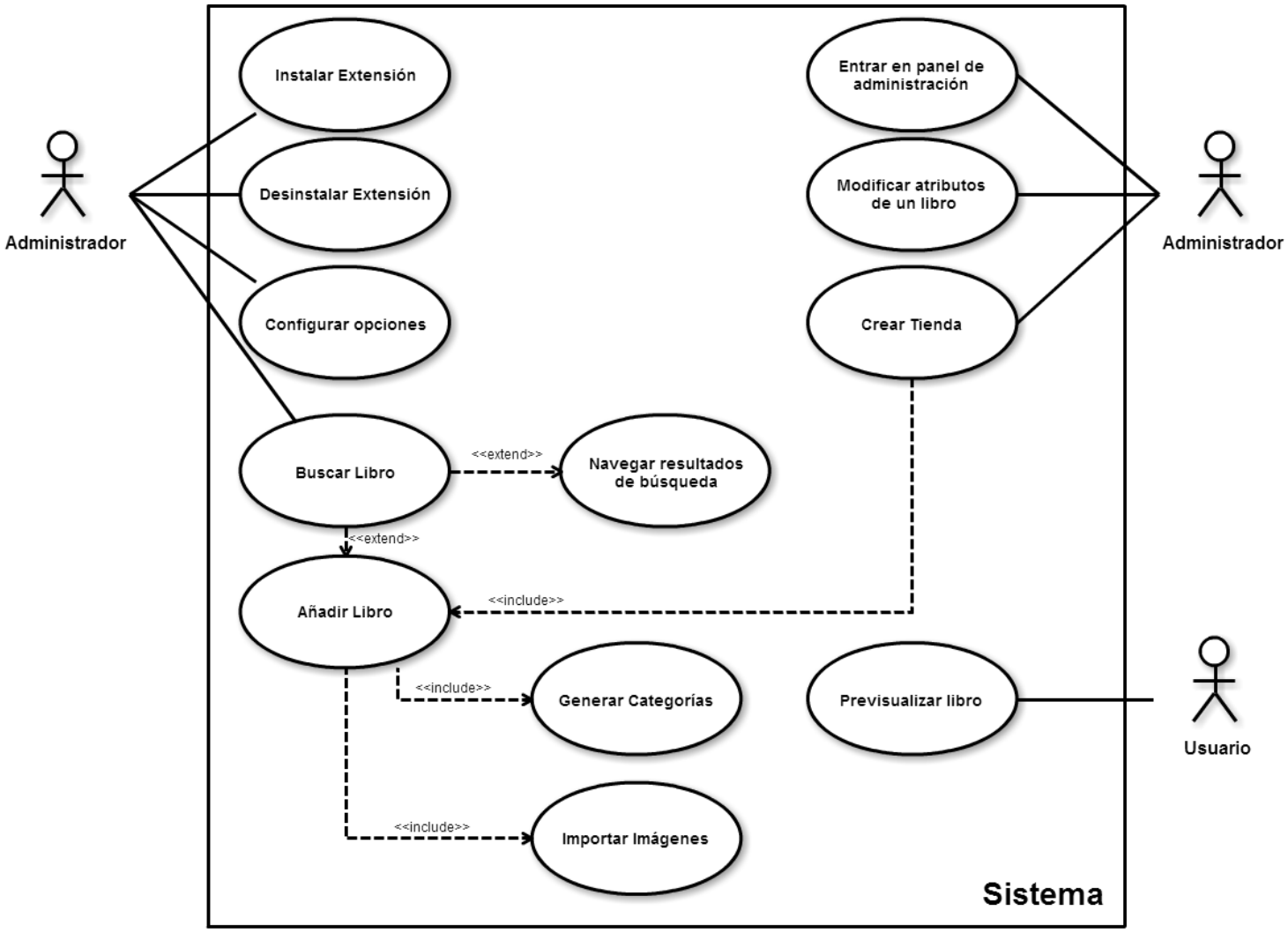

Figura 13. Diagrama de casos de uso. Nivel 1.

- **Instalar extensión:** acción de incorporar la extensión a la lista de plugins instalados poniendo a disposición de los usuarios administradores la funcionalidad que proporciona
- **Desinstalar extensión:** retirar la extensión del listado de plugins disponibles para uso de los administradores
- **Configurar opciones:** especificar los parámetros de configuración de la extensión

- **Buscar libro:** acción de solicitar un listado de libros a Google Books mediante la funcionalidad de búsqueda de Books API
- **Añadir libro**: agregar un producto a NopCommerce desde el listado de resultados de búsqueda
- **Generar categorías:** agregar las nuevas categorías de los productos agregados que aún no existan en NopCommerce
- **Importar imágenes:** agregar las imágenes de los productos provenientes de Google Books como imágenes de producto en NopCommerce
- **Modificar atributos de un libro:** actualizar en NopCommerce la información de un libro proporcionada por Google Books
- **Crear tienda:** acción por la que se genera un nuevo portal de libros mediante la página construida al efecto
- **Pre-visualizar libro:** abrir el visor de libros desde la ficha de producto por parte de un usuario

3.3. Diagrama de clases

En este apartado se muestra un diagrama de clases a muy alto nivel sobre cómo se relacionan las clases de MVC con sus principales propiedades y métodos, para el ámbito de la extensión desarrollada:

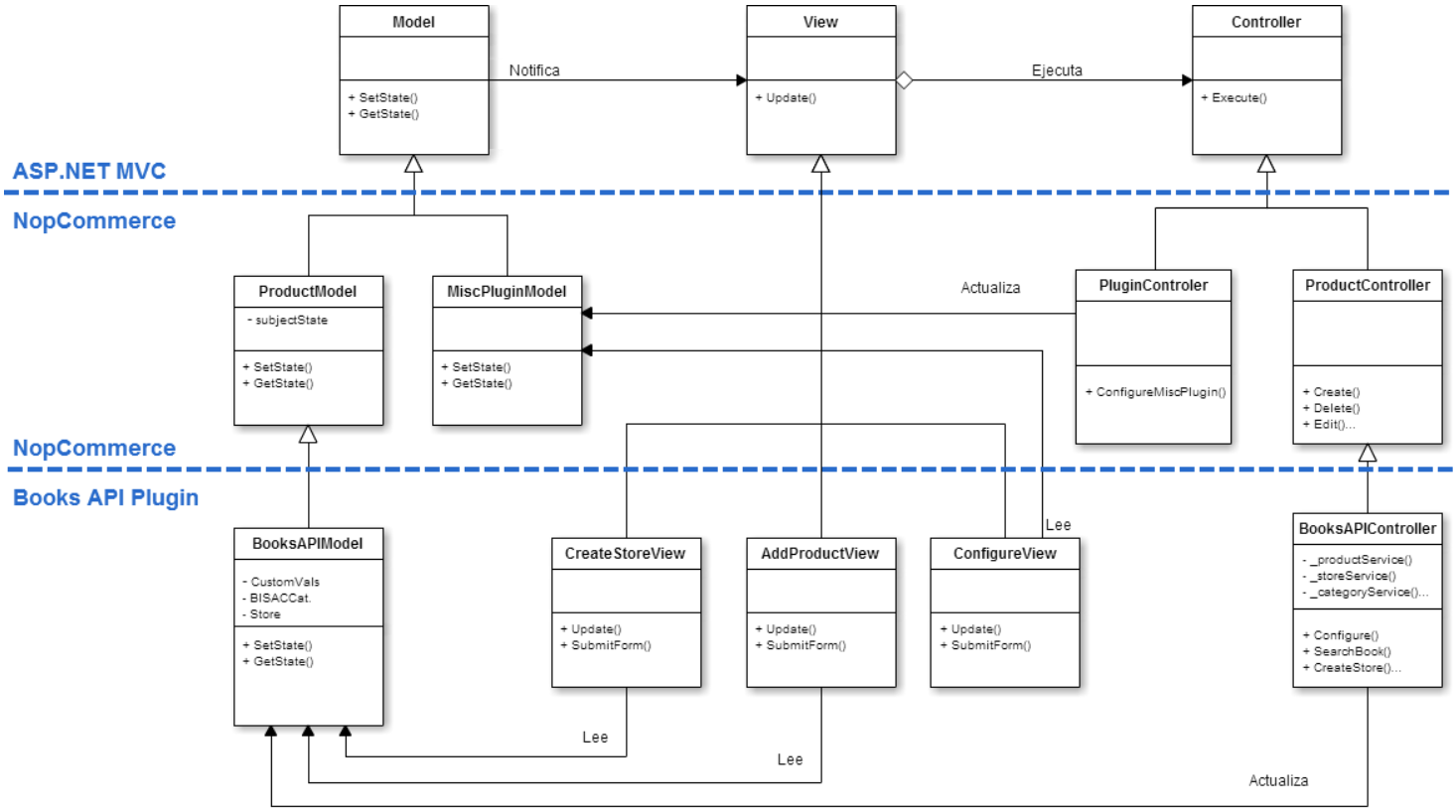

Figura 14. Diagrama de clases.

Se puede ver de qué se encarga cada clase. Empezaremos con los modelos. El modelo, en la implementación que Microsoft ha hecho de MVC con ASP.Net, se encarga de notificar a la vista que el modelo ha cambiado. Por tanto, internamente almacena un contexto con el estado actual de sus propiedades, y que la vista consulta cuando esta le notifica que ha cambiado:

- **ProductModel**: es la clase que representa el modelo producto en todo NopCommerce, y del que hemos heredado mediante la clase BooksAPIModel para beneficiarnos de todas las propiedades y métodos desarrollados para gestionar productos.
- **BooksAPIModel**: es el modelo principal de la extensión y hereda de ProductModel, que podríamos decir que es el modelo principal de NopCommerce. Se han agregado algunas propiedades importantes, como "CustomVals" que es una nueva subclase donde están contenidas las

propiedades de producto que vienen de Google Books y que no están en la clase ProductModel de NopCommerce. En "CustomVals" se puede encontrar propiedades como "*Authors*", "PageCount", "*Title*" o "WebReaderLink", que proporciona la URL con la que construir el enlace para pre-visualizar cada libro.

A continuación, se detallan las vistas que representan la parte de presentación que el servidor utiliza para construir el HTML que devuelve al navegador. Las vistas consultan el modelo asociado cada vez que este les notifica que ha cambiado, de manera que se encuentre siempre actualizado.

- **BooksAPIView**: son el conjunto de vistas que conforman la extensión, y que son las vistas "ProductExtensionTab" utilizada para presentar los valores personalizados "CustomVals" detallados en el modelo "BooksAPIModel" en la vista propia de NopCommerce para administrar las propiedades de un producto, "CreateStore" que muestra y envía el formulario donde se rellenan los datos de la tienda a crear y "AddProduct", que se encarga de mostrar el formulario de búsqueda de libros y los resultados que devuelve Google Books API.
- **ConfigurationView**: es la vista que muestra el formulario para modificar la configuración de cualquier extensión. El modelo que tiene asociado es MiscPluginModel, que NopCommerce proporciona para implementar la configuración de las extensiones.

Por último, los controladores son los que se ocupan de ejecutar la lógica de negocio, y disponen de métodos que devuelven el modelo resultante de aplicar las operaciones correspondientes:

- **BooksAPIControler**: contiene la lógica para buscar un libro contra el directorio de Google Books, añadirlo a NopCommerce, crear una tienda con las categorías seleccionadas por el administrador y editar los valores personalizados.
- **PluginController**: ejecuta las operaciones de grabado de la nueva configuración en la base de datos específicamente para el *plugin* seleccionado, la extensión Books API.

3.4. Diagramas de flujo

Los diagramas de flujo son una buena herramienta para comprender mejor cómo discurre la ejecución del programa, y es de gran utilidad en la fase de análisis para acordar con el usuario final cuál ha de ser el orden de ejecución, y para el equipo programador, cómo disponer los módulos de la lógica de negocio. Suelen detallarse qué validaciones se van a efectuar, qué procesos deben lanzarse, qué *inputs* deben realizarse, qué entradas de fichero se esperan en cada proceso, etc. En este apartado mostraremos el flujo para los casos de configurar la extensión, buscar un libro, y crear una tienda al más alto nivel.

- Configuración: es el proceso por el cual el usuario introduce los parámetros de personalización de la extensión, recibe la confirmación de que la validación de los datos insertados es correcta y son guardados por el sistema. Caso de uso relacionado: "Configurar opciones".

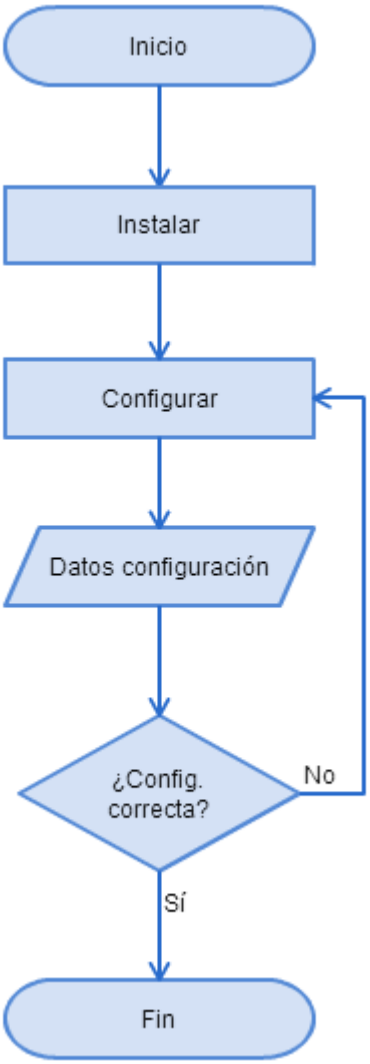

Figura 15. Diagrama de flujo. Configuración.

- Buscar un libro: el administrador introduce una cadena de búsqueda, el sistema busca en Google Books, el sistema recibe una respuesta, formato JSON, y el administrador pincha en "*Add*" para añadir un libro.

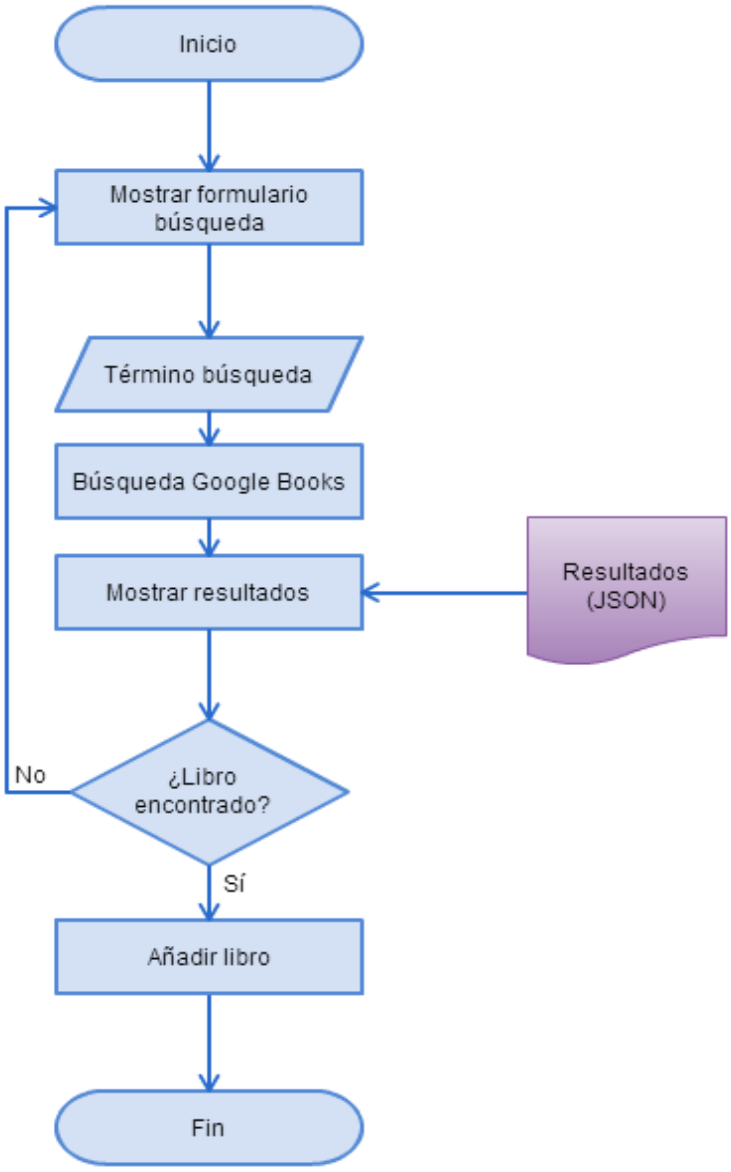

Figura 16. Diagrama de flujo. Buscar un libro.

Casos de uso relacionados: "Buscar libro", "Añadir libro", "Generar Categorías" e "Importar Imágenes"

- Crear una tienda: el administrador introduce los datos de creación de la tienda, el sistema agrega los productos resultantes según la búsqueda en Google Books por las categorías seleccionadas que el sistema recibe en formato JSON.

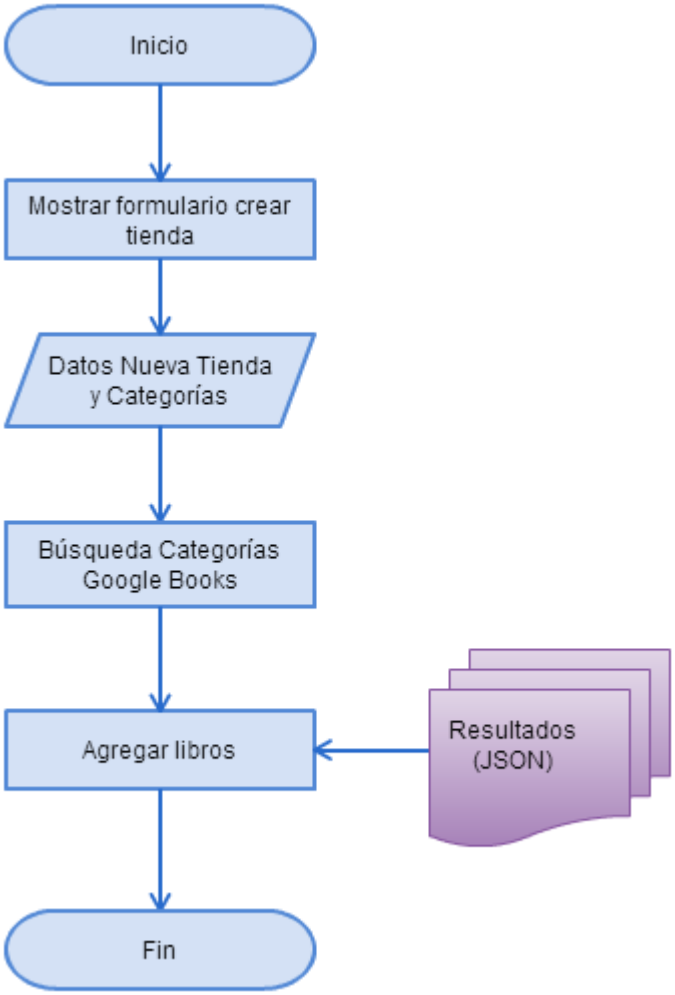

Figura 17. Diagrama de flujo. Crear una tienda.

Casos de uso relacionados: “Crear Tienda”

4. Diseño

La fase de diseño, según Taylor (1959), es el proceso de aplicar distintas técnicas y principios con el propósito de definir un dispositivo, proceso o sistema con los suficientes detalles como para permitir su realización física. A continuación detallamos la arquitectura del sistema y el modelo Entidad/Relación asociados a la aplicación del presente proyecto.

4.1. Arquitectura del sistema

La arquitectura de software está compuesto por un conjunto de patrones y abstracciones utilizados para la construcción del software en un sistema de información. Define qué componentes conforman el sistema y cómo estos se comunican entre sí.

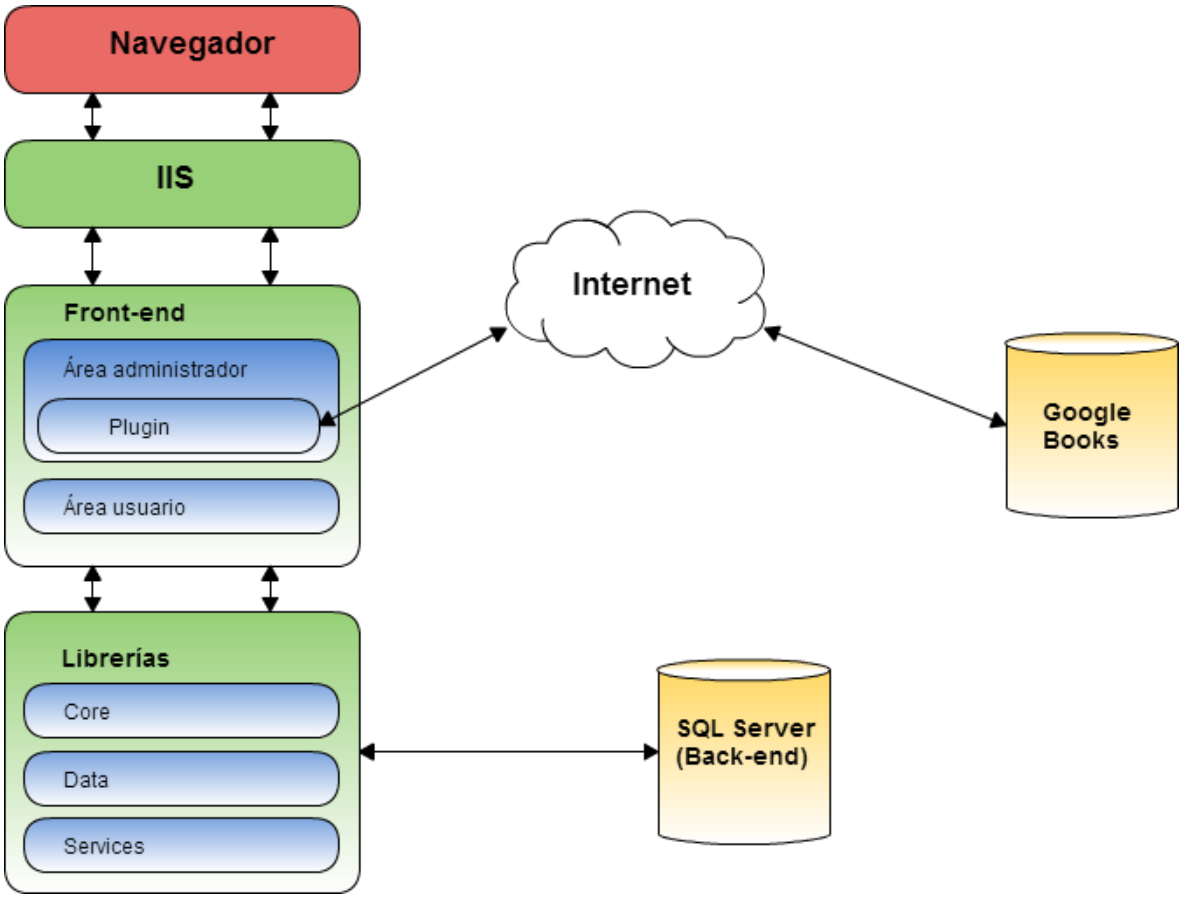

Figura 18. Arquitectura del sistema.

Como se puede ver en la figura 18, el navegador web es quien inicia la interacción en el sistema, que transmite una petición al servidor web IIS. El servidor le pasa la llamada a la capa de presentación de la aplicación (*front-end*), que se ocupa de procesar la petición y haciendo a su vez las llamadas necesarias a la capa de negocio (librerías).

La capa de presentación, está compuesta a su vez por dos módulos bien diferenciados, que se ocupan de manejar la aplicación web del usuario que navega, y la del administrador del *e-commerce*. Los *plugins* son parte de este último módulo, aunque en la solución de Visual Studio esté separado como otro proyecto.

Una vez procesada, el *front-end* le devuelve el HTML resultante al servidor web, que lo envía al navegador para su visualización.

Existe la interacción particular de la extensión con el servidor de Google Books para recopilar los datos de libros, que Google Books sirve en formato JSON, formato muy conocido entre los programadores de *webservices*, y uno de los estándares de envío de datos en comunicaciones entre aplicaciones conectadas a Internet más utilizados.

4.2. Modelo Entidad/Relación

Cuando se necesita crear un modelo de datos en un sistema de información, se utilizan herramientas que puedan, mediante una abstracción, definir cómo será el diseño de la base de datos en el momento de la implementación.

Las entidades son la representación en el modelo de cualquier concepto del mundo real que ayudarán a entender mejor el problema y cómo crear tablas y sus correspondientes campos o atributos una vez se construya la base de datos.

En el presente caso se podría crear el diagrama del modelo de todo el sistema NopCommerce, pero se va a mostrar el diagrama correspondiente a las entidades relacionadas con la extensión del proyecto a desarrollar.

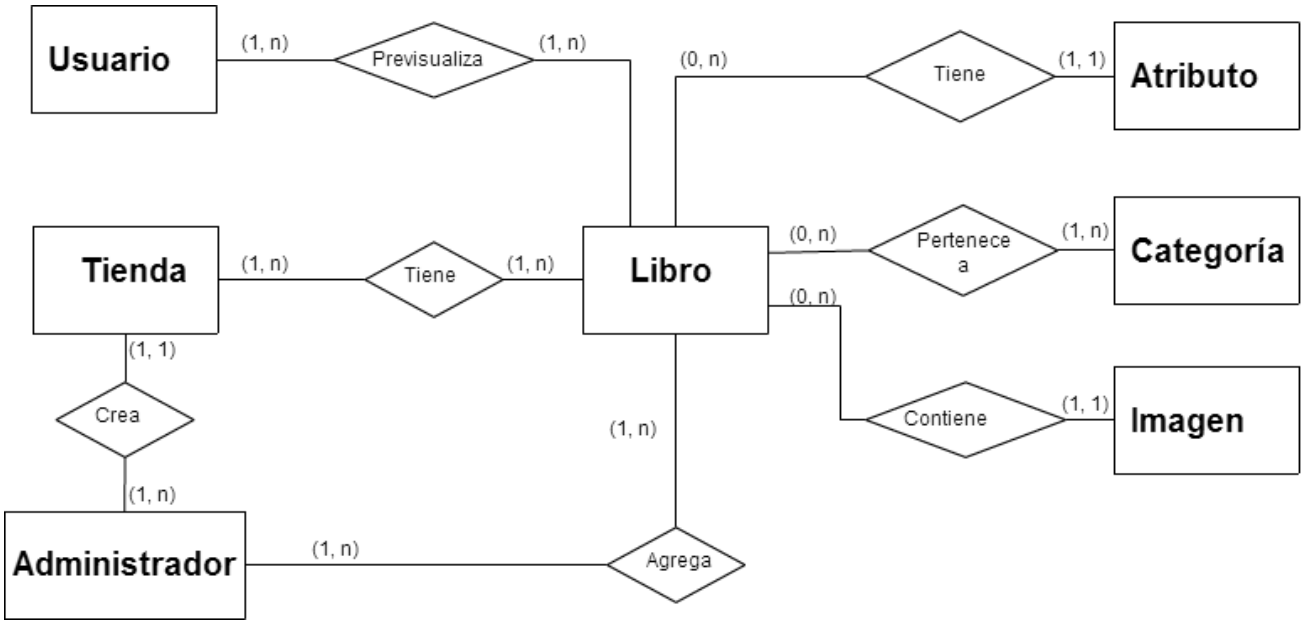

Figura 19. Modelo Entidad/Relación.

A continuación se describen brevemente qué representa cada entidad y sus respectivas relaciones:

- **Usuario**: es la persona que accede a la tienda con la finalidad de comprar o de realizar cualquier otra operación restringida al rol de usuario. En el caso de este programa, un usuario puede pre visualizar a través del API de pre visualización cada uno de los libros de la tienda. El resto de relaciones con las entidades de que pueda constar el sistema visto en su generalidad no se contempla en este diagrama.
- **Administrador:** es la persona que accede a la tienda para gestionar la información relativa al *e-commerce*, y utilizar el área restringida a este rol. Entre los permisos con que cuenta, incluye el de agregar libros y crear tiendas que vienen reflejados en el diagrama propuesto.
- **Tienda:** cada una de las librerías que se crean por parte del administrador y que contiene los libros agregados en el momento de crear la tienda o por el administrador en un momento puntual. Cada libro puede publicarse en más de una tienda.

- **Libro:** cada entidad única de libro es incorporado al sistema en el proceso de creación de una tienda al estar contenido en una de las categorías seleccionadas por el administrador, o la búsqueda y posterior adición de forma unitaria también por parte del administrador. Se han definido como entidades relacionadas, el atributo, la categoría y la imagen (que provienen de Google Books pero se insertarán en la base de datos).
- **Atributo:** al consultar la información del libro en Google Books, existen algunos atributos que no se encuentran en NopCommerce y por tanto es necesario agregarlos al sistema para poder mostrarlos o utilizarlos.
- **Categoría**: las categorías del estándar BISAC son agregadas a la base de datos al instalar el programa, y se utilizan para dar la posibilidad al administrador de seleccionar aquellas categorías con las que buscar los libros a adicionar a la tienda que se va a crear. Como se ha comentado anteriormente, Google Books utiliza este estándar para categorizar sus libros, y el presente programa también lo hace.
- **Imagen:** para poder mostrar las imágenes de un libro en la ficha de cliente primero debe insertarse en la base de datos. El programa las descarga en el momento de hacer la consulta de un libro a Google Books y las graba en la base de datos. Un libro puede contar con varias imágenes según la resolución.

5. Implementación

La fase de implementación en Ingeniería del Software es aquella en la que se construye el componente de software que satisface los requerimientos enunciados en la fase de análisis, utilizando para ello herramientas de ayuda a la programación, lenguajes de programación y algoritmos.

En el presente capítulo se explica en qué consiste esta fase en el proyecto que se está tratando, qué módulos componen el programa, el detalle de los algoritmos desarrollados, y cómo programarlos.

5.1. Organización de la solución

La organización del proyecto (ver figura 20) en la herramienta utilizada para el desarrollo, Visual Studio 2013, atiende a dos metodologías utilizadas para desarrollar el programa: MVC y *Entity Model*. En el caso de *Entity Model*, se utiliza puntualmente para incorporar la entidad de categorías "BISACCategory[48]" al *e-commerce*, y de esta manera se evita modificar nada del proyecto original de NopCommerce.

[48] Ver capítulo 2.5. de la memoria

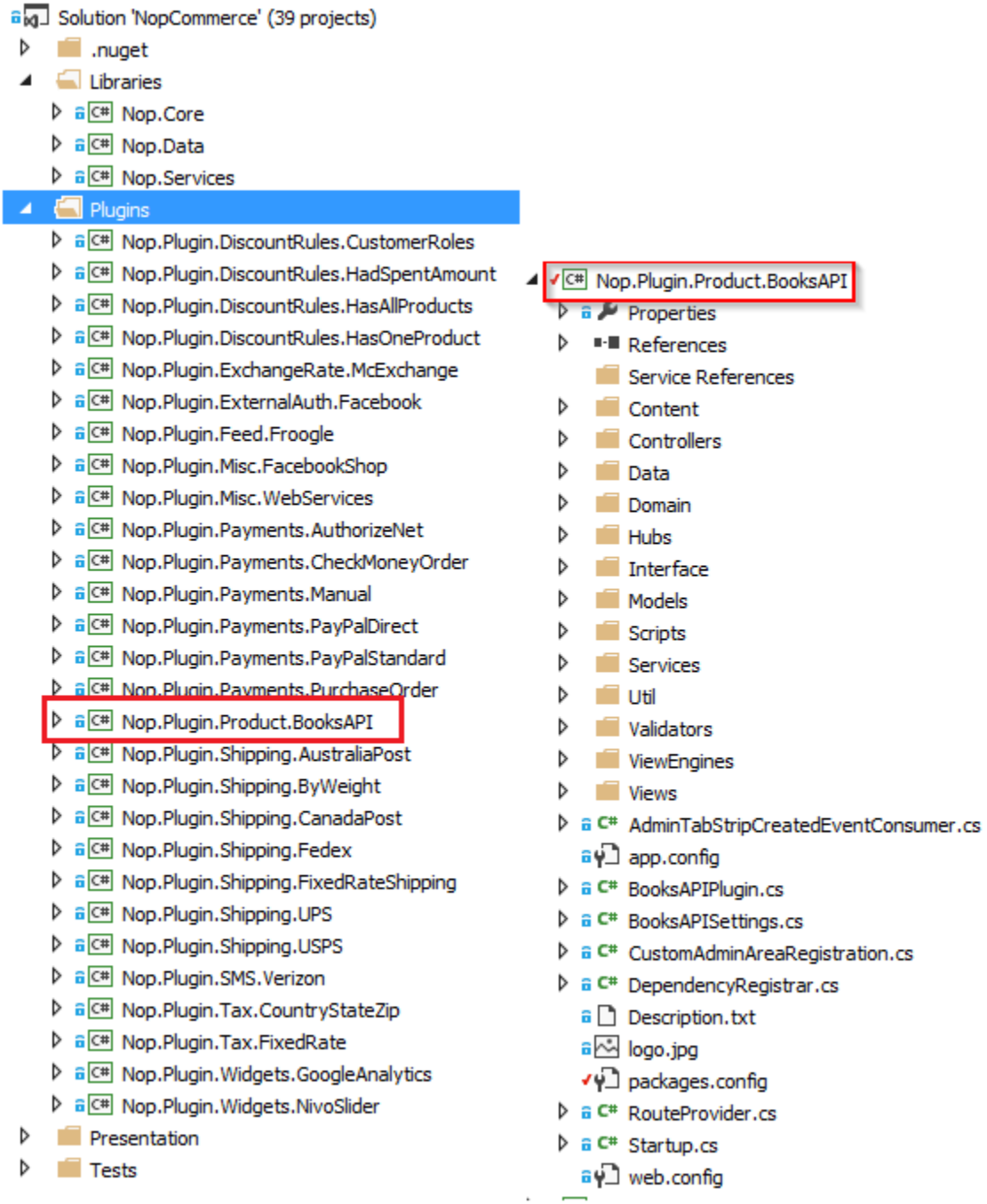

Figura 20. Organización del proyecto.

Algunas de las carpetas que existen, vienen pre definidas por Visual Studio, como son:

- ***Properties*:** guarda las propiedades del ensamblado resultante. Aquí se puede especificar, por ejemplo, un GUID específico para una librería de funciones cuando es referenciada como objeto COM.
- ***References*:** aquí se pueden encontrar todas las referencias a librerías dinámicas que se han incorporado al proyecto para utilizar funcionalidad de componentes externos, como por ejemplo el *Entity Framework*[49]o el *Automapper*[50].
- ***Service References***: almacena los vínculos a web services para que sean consumidos utilizando SOAP o WCF. Como no es el caso, esta carpeta figura vacía.
- ***Content***: contiene todos los recursos estáticos del proyecto, como pueden ser imágenes, logos y hojas de estilo.

El resto, son carpetas que se han ido creando a lo largo del desarrollo y según iba siendo necesario.

A continuación, se detalla el contenido de dichas carpetas:

- ***Controllers***: aquí se encuentran las clases controlador, que juegan un papel fundamental en el patrón de desarrollo MVC, pues se encargan de ejecutar toda la lógica asociada a un modelo para actualizarlo. El único controlador que ha sido necesario crear, y que se explicará posteriormente, es **BooksAPIController**, el controlador asociado al modelo creado para manejar la información que se obtiene de Google Books.
- ***Data***: contiene todo lo relacionado con la capa de datos que haya que incluir en el *plugin*. En un proyecto al uso, en el que toda la aplicación se construye desde el principio, esta carpeta no se habría encontrado en un proyecto aislado de la carpeta *Data* de la aplicación. Pero se ha

[49] https://msdn.microsoft.com/es-es/data/ef.aspx
[50] https://automapper.codeplex.com/

de recordar que es una extensión, y como tal no debe requerir modificar nada del proyecto de la aplicación original. Para la entidad BISACCategory, nos era necesario crear estas clases dentro de la carpeta *Data*, y es por eso que se ha creado en el proyecto del *plugin* en lugar de en la carpeta "*Libraries/Data*" de NopCommerce.

- ***Domain*:** contiene las entidades del dominio a definir. En este caso, únicamente se cuenta con BISACCategoryRecord, que representa la entidad "BISACCategory".
- ***Hubs*:** esta carpeta la utiliza el *framework* "SignalR" para crear los proxies desde los cuales comunicarse con los clientes. Se ha creado "ProgressHub" que se utiliza para enviar mensajes a la ventana que muestra el progreso de crear una tienda.
- ***Interface*:** son las interfaces del Entity Model, en este caso únicamente se necesita una para el servicio de la entidad "BISACCategory".
- ***Models*:** contienen los modelos a utilizar en la aplicación. Son clases que publican propiedades que el MVC *Framework* definirá como sus modelos. En el caso del presente proyecto se hicieron necesarios 3 modelos: BooksAPI, BISACCategory y *Configuration*.
- ***Scripts*:** aquí se guardan todos los scripts de Javascript y JQuery que se necesitan en el proyecto. Concretamente, se incluyen los scripts relativos a JQuery, JQuery UI, JQuery Multiple Select, un *plugin* para mostrar los "*multi select*" html de una manera más atractiva, como se puede ver en la pantalla de "Crear tienda", y JQuery SignalR, que implementa las operaciones necesarias para hacer peticiones desde el navegador al módulo servidor de SignalR.
- ***Services*:** son los servicios del Entity Model cuyas interfaces se han publicado en la carpeta "Interface".
- ***Util*:** aquí se encuentran las clases que definen procedimientos y funciones que no son propios de los modelos. Está la clase "IISAdmin" que nos posibilita manejar el servidor Web IIS de manera programática. En este caso concreto, se ha utilizado para crear los enlaces Host correspondientes a cada tienda creada. Un enlace Host es la URL en la que responde un sitio web. De tal manera, en IIS se

pueden tener definidos sitios web para el mismo puerto (el 80, por ejemplo) pero con enlaces distintos (ver figura 21).

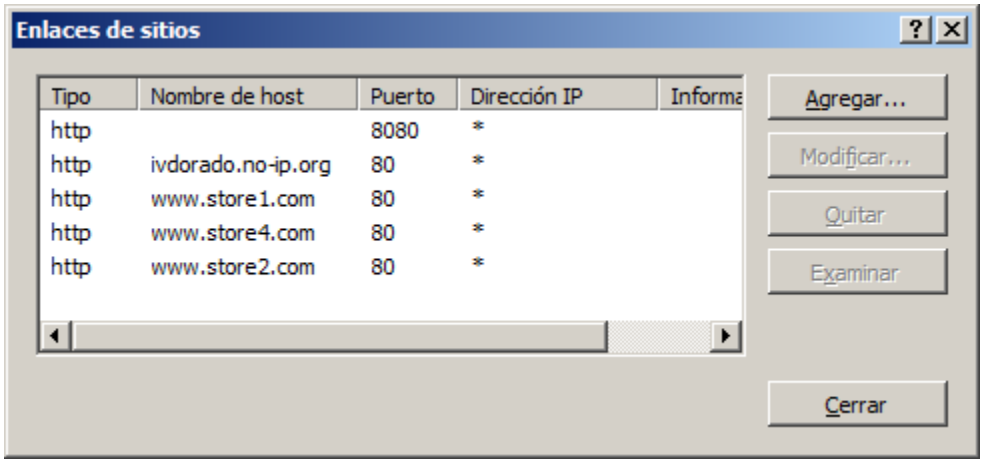

Figura 21. Enlaces en IIS.

- ***Validators***: contiene las clases que servirán para hacer la validación de los datos introducidos por el usuario.
- ***ViewEngines***: debido a que el motor de búsqueda por defecto de MVC *Framework*, busca las vistas en ciertos directorios del proyecto, para que encontrara las vistas y así poder sobrecargar las existentes en el proyecto principal, se tuvo que crear un nuevo motor de búsqueda de vistas que herede el motor de MVC, para así poder insertar en la lista de ubicaciones las de las vistas de la presente extensión. [5] [6] [9]
- ***Views*:** son las vistas que componen el patrón modelo-vista-controlador, y que sirve para decirle al servidor web cómo debe mostrar los datos del modelo. Se asocia con la capa de presentación. En este proyecto se cuenta con las vistas del modelo BooksAPI, que son:
    - ***_ProductExtensionTab*:** vista parcial para inyectar el formulario de edición de los valores customizados de producto

que vienen de Google Books en el menú de gestión de producto. [5] [6]

- ***AddProduct*:** vista para mostrar el formulario de búsqueda de producto y adición desde los resultados de búsqueda que devuelve Google Books.
- ***Configure*:** muestra el formulario de configuración de la extensión.
- ***CreateStore*:** muestra el formulario para completar la creación de una tienda.

Las vistas del modelo Producto (perteneciente al proyecto principal que se sobrecargan desde la extensión):

- ***_ProductCustomDetails***: vista parcial que se inyecta en la vista de ficha de producto del área de usuario:

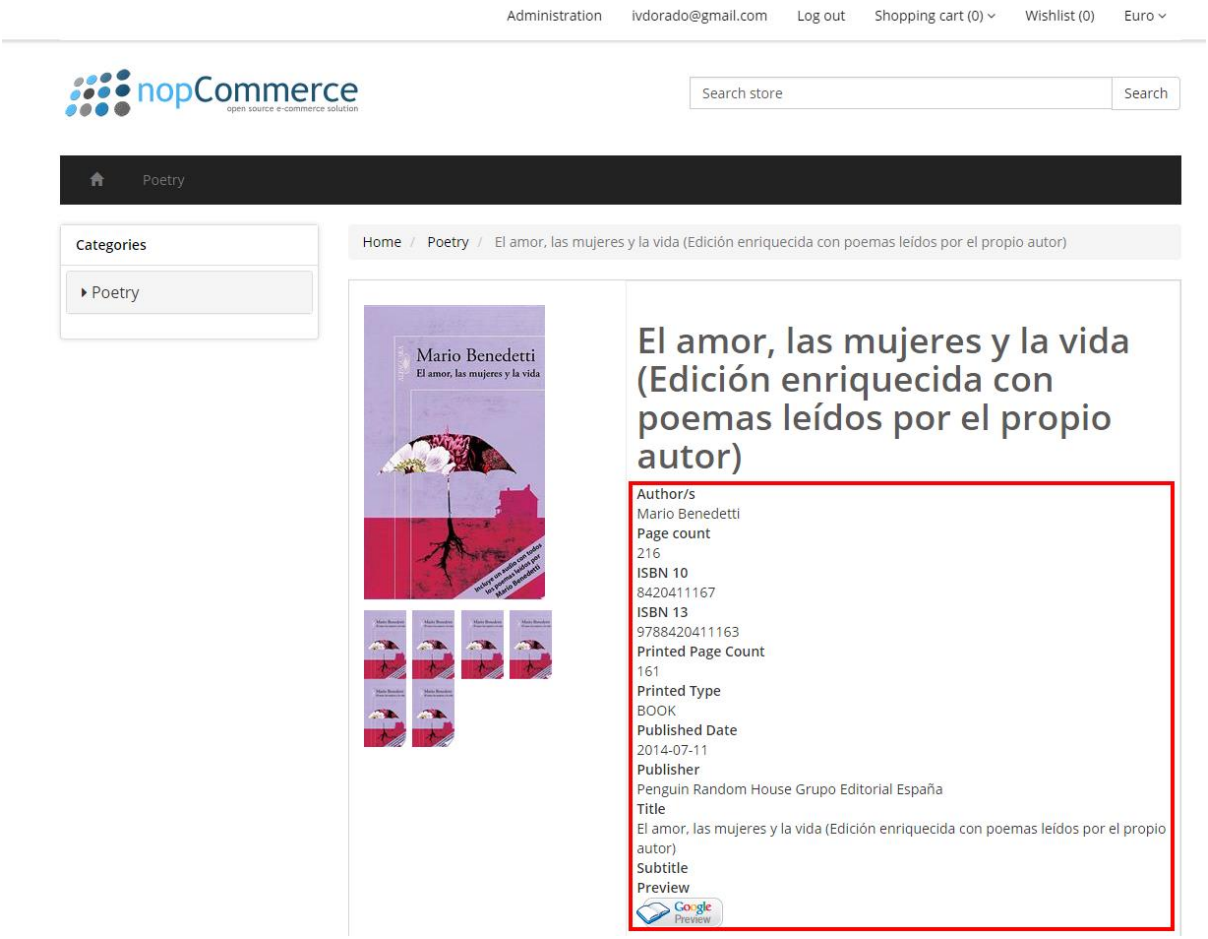

Figura 22. Vista parcial donde mostramos los datos obtenidos de Google Books.

- ***ProductTemplate.Simple***: vista sobre cargada de la perteneciente al proyecto principal. Aquí se inyecta la vista parcial *_ProductCustomDetails*.

- **Directorio raíz**: se encuentran las clases definidas para otros fines en la extensión, como son:
  - ***AdminTabStripCreateEventConsumer***: controla el evento de creación de las pestañas de la página de gestión de producto, ya que servirá para agregar la pestaña donde se podrán editar los valores customizados provenientes de Google Books.
  - **BooksAPIPlugin**: define los métodos de instalación y desinstalación del *plugin*, así como los valores que tendrán

cada uno de los ítems del menú "*Plugin* -> BooksAPI" del menú en el sitio de administración.

- **BooksAPISettings**: define qué valores conformarán la configuración de la extensión.
- ***CustomAdminAreaRegistration***: agrega la ruta a la vista de edición de valores customizados provenientes de Google Books. [5] [6]
- ***DependencyRegistrar***: registra los tipos del Entity Model para que estén disponibles en el MVC *Framework*.
- ***Description***.txt: de utilidad para NopCommerce ya que muestra esta información en el listado de extensiones disponibles.
- ***RouteProvider***: registra todas las rutas del *plugin* para puedan ser llamadas mediante la URL acortada http://url_del_sitio/BooksAPI/vista.
- **Startup**: define las operaciones a ejecutar cuando se inicia la aplicación.

5.2. Detalle de las clases principales

A continuación, se pasa a dar el detalle de los algoritmos implementados en las principales clases que componen la solución. Es una manera de acercar al lector a las rutinas utilizadas y las técnicas desarrolladas.

5.2.1. Modelo

El modelo es la representación de la información dentro del patrón MVC, y generalmente se construye únicamente definiendo qué datos van a intervenir, y en el constructor qué subclases se van a instanciar si es que las hay. Se debe intentar despojar al Modelo de toda lógica de negocio. Las operaciones sobre el Modelo, conceptualmente, las realiza el controlador. Por tanto, un Modelo es la parte más sencilla a la hora de construir una aplicación MVC.

El modelo del que se va a ver un extracto a continuación, es el que se ha utilizado como Modelo principal de nuestra extensión, `BooksAPIModel`. Es una herencia del Modelo estándar de nopCommerce `ProductModel`.

5.2.3.1. Definición de atributos

Básicamente, son los miembros del Modelo, los que se necesita después rellenar con el controlador:

```
public string ProductSearchName { get; set; }
public string ProductSearchIsbn { get; set; }
public string ProductSearchSubject { get; set; }
public int SelectedBISACCategory { get; set; }
[NopResourceDisplayName("Plugin.Product.BooksAPI.Categories")]
public IEnumerable<string> MultiSelectedBISACCategories { get; set; }
public IEnumerable<SelectListItem> BISACCategories { get; set; }
public CustomValues CustomVals { get; set; }
```

[…]

5.2.3.2. Constructor

En el caso concreto de `BooksAPIModel`, se le pasa un volumen, `Volume`, de Google Books, con el que instanciar el modelo. Dado que el modelo está concebido para representar los datos que vienen de Google Books, pero la clase `Volume` es externa al sistema, se hace aquí un *mapeo* de la estructura:

```
public BooksAPIModel(Volume volume)
 {
     //Unique ID
     Sku = volume.Id;
     //Name of the product is the title of the book
     Name = volume.VolumeInfo.Title;
     //Short description is the subtitle
     ShortDescription = volume.VolumeInfo.Subtitle;
     //Full description is the Description of the book
     FullDescription = volume.VolumeInfo.Description;
```

[…]

5.2.3.3. Subclases

En el Modelo que se está tratando, se ha definido una subclase utilizada para representar aquellos valores de Google Books que no están definidos en la clase original de nopCommerce `ProductModel`. Se llama `CustomValues`.

```
public CustomValues(Volume volume)
 {
     string authorsStr = String.Empty;
     if (volume.VolumeInfo.Authors != null)
     {
         for (int i = 0; i < volume.VolumeInfo.Authors.Count; i++)
         {
          if (i == 0) { authorsStr = volume.VolumeInfo.Authors[i]; }
          else { authorsStr = authorsStr + ", " +
          volume.VolumeInfo.Authors[i]; }
         }
     }
     Authors = authorsStr;
     PageCount = (int)volume.VolumeInfo.PageCount;
   […]
```

5.2.2. Vista

A continuación, se esboza brevemente, cómo se construye la vista correspondiente a un modelo, en este caso se muestra un extracto de la vista "AddProduct" que nos sirve para buscar y agregar libros.

La sintaxis empleada es la llamada Razor, que es un lenguaje de marcado que permite encrustar código Visual Basic o C# en HTML. Esto permite generar contenido HTML dinámico en el servidor, al estilo de PHP, ASP, JSP… como resultado de operaciones de la lógica de negocio de la aplicación y del acceso a datos de un *backend*.

Al igual que en una clase de C#, debemos incluir las librerías externas de las que se vayan a utilizar recursos, mediante la cláusula `@using`:

```
@using Nop.Core.Infrastructure;
@using Nop.Web.Framework.UI;
@{
    var defaultGridPageSize =
EngineContext.Current.Resolve<Nop.Core.Domain.Common.AdminAreaSettings>().DefaultGridP
ageSize;
```

```
    var gridPageSizes =
EngineContext.Current.Resolve<Nop.Core.Domain.Common.AdminAreaSettings>().GridPageSize
s;

    Layout = "~/Views/Shared/_AdminLayout.cshtml";

    Html.AddCssFileParts("~/Plugins/Product.BooksAPI/Content/Styles/Grid.css");
}
@model Nop.Plugin.Product.BooksAPI.Models.BooksAPIModel
@using Nop.Web.Framework;
```

La línea más interesante en el bloque de código anterior, que es el inicio de la vista, es la declaración del modelo que utiliza la vista:

```
@model Nop.Plugin.Product.BooksAPI.Models.BooksAPIModel
```

El MVC *Framework* se ocupa de todas las operaciones de consulta del modelo, y actualizaciones de la vista.

Obviamente, las vistas Razor, también permiten la ejecución de código en el cliente mediante Javascript o JQuery. En el caso de la vista que se va a mostrar, se utiliza un *plugin* de JQuery llamado "Kendo Grid[51]" que permite formatear listas en forma tabular, paginada y con otras utilidades, de manera muy sencilla, como este bloque, que inicializa el parámetro de origen de datos del Grid:

```
var selectedIds = [];

    $(document).ready(function () {

        //search button
        $('#search-products').click(function () {
            //search
            $("#products-grid").kendoGrid({
                autobind: false,
                dataSource: {
                     type: "json",
                     transport: {
                            read: {
                            url: "@Html.Raw(Url.Action("SearchBook", "BooksAPI"))",
                            type: "POST",
                            dataType: "json",
                             data: additionalData
                        }
```

[51] http://demos.telerik.com/kendo-ui/grid/index

```
},
```

Los *inputs* de datos tienen este aspecto, que como se puede ver llaman directamente a un atributo del modelo:

```
<tr>
     <td class="adminTitle">
         <label for="SearchInput">@T("Admin.Common.Search")</label>
     </td>
     <td>
         @Html.EditorFor(model => model.ProductSearchName)
     </td>
</tr>
```

No se va a entrar en mucho más detalle sobre qué son los editores como el invocado mediante `@Html.EditorFor`, o lo que son las expresiones "lambda", como `model => model.ProductSearchName`, pero sí se proporcionan dos referencias [52] [53] donde poder indagar un poco más sobre cómo funcionan.

5.2.3. Controlador

El controlador es la parte del programa que ejecuta la lógica para actualizar el modelo que está mostrando la vista.

Los controladores son clases predefinidas del MVC *Framework*, por lo que al construir nuestro propio controlador, heredaremos la clase `Controller` de MVC.

Necesitamos disponibles todos los servicios o repositorios creados en el dominio del Entity Model, por lo que definiremos variables locales que reciben estos servicios en el constructor.

Después, en cada acción, se implementa qué operaciones realizar sobre el modelo para devolverlo incluyendo los cálculos que se requerían.

[52] Razor Markup – What is Razor: http://www.w3schools.com/aspnet/razor_intro.asp
[53] Understand Lambda Expressions in 3 minutes: http://www.codeproject.com/Tips/298963/Understand-Lambda-Expressions-in-minutes

Normalmente, para cada acción se crea una función para la llamada "*get*" y otra para la llamada "*post*". Esto se especifica como un atributo encima de la función que define cada acción. De este modo se ejecuta código distinto cuando la llamada se hace mediante cada una de las distintas formas de petición HTTP.

Ejemplo, método *GET*:

```
[HttpGet]
 public ActionResult SearchBook()
 {
     BooksAPIModel model = new BooksAPIModel
     {
         //Fill in the categories from the BISAC entity
         BISACCategories = _bisacCategoryService.GetBISACCategoriesSelectList()
     };
     //Return model to "AddProduct" view
     return View("AddProduct", model);
  }
```

Ejemplo, método *POST*:

```
[HttpPost]
public ActionResult SearchBook(BooksAPIModel model, DataSourceRequest command)
{
  var gridModel = new DataSourceResult();
  var storeScope = this.GetActiveStoreScopeConfiguration(_storeService, _workContext);
  var booksAPISettings = _settingService.LoadSetting<BooksAPISettings>(storeScope);
...
```

El controlador `BooksAPIController`, que hace de controlador para el *plugin*, contiene algunas funciones que se deben detallar un poco para saber qué hacen exactamente.

Las principales, son las que se ocupan de buscar un libro, crear una tienda, y configurar la extensión. "`SearchBook`" es llamada cuando se llama a la ruta "/BooksAPI/SearchBook", y la explicación sencilla es porque en la clase "RouteProvider", que extiende la existente en el proyecto base, asigna a estas vistas la acción correspondiente como se ve aquí:

```
//Route for searching a book
```

```
    routes.MapRoute("Plugin.Product.BooksAPI.SearchBook",
"Plugins/Product/BooksAPI/SearchBook",
                new { controller = "BooksAPI", action = "SearchBook", area="Admin" },
                          new[] { "Nop.Plugin.Product.BooksAPI.Controllers" });
```

Este "mapeo" se establece dentro de la rutina "`public void RegisterRoutes(RouteCollection routes)`" que recibe el objeto routes al que añade las rutas que se estimen importantes. Desde ese momento, la ruta especificada, responde a la acción del controlador pertinente.

5.2.1.1. Buscar libro

Lo primero es fijarnos que en la cabecera de la acción tipo "POST", recibida mediante Ajax desde el formulario mostrado por la vista "AddProduct.cshtml", está el modelo, y un objeto del tipo `DataSourceRequest:`

```
[HttpPost]
public ActionResult SearchBook(BooksAPIModel model, DataSourceRequest command)
```

En el modelo, únicamente se obtienen valores en los campos de búsqueda publicados en el formulario.

Primero se comprueba que no vengan vacías:

```
 //Verify the input search is not empty
if (model.ProductSearchName != "" && model.ProductSearchName != null)
{
    //Create the query string
    var q = model.ProductSearchName;
    //Verify the subject input is not empty
    if (model.ProductSearchSubject != null)
    {
        //Add the subject parameter to the query string
       q = q + "+subject:" + model.ProductSearchSubject;
    }
```

En tal caso se construye, en la variable "q" de tipo cadena, la cadena de búsqueda que se quiere enviar a Google Books API.

Acto seguido, se crea el objeto de la API de Google Books que se encarga de gestionar las llamadas, `VolumesResource.ListRequest`, y se le pasa la cadena de búsqueda, "q". Posteriormente se configura el objeto y se ejecuta la búsqueda:

```
//The object ListRequest is created based on the query string built
VolumesResource.ListRequest list = _booksService.Volumes.List(q);
//Set the correct parameters for the search query
list.OrderBy =
(VolumesResource.ListRequest.OrderByEnum)booksAPISettings.SelectedOrderBy;
list.LangRestrict = booksAPISettings.SelectedBookLanguage == 1 ? "es" : "en";
list.MaxResults = command.PageSize;
list.StartIndex = (command.Page - 1) * command.PageSize;
//Execute the search
Volumes bookVolumes = list.Execute();
```

Los resultados se pasan a la variable "gridModel" que muestra los valores cargados en el grid de Kendo. Para ello, se necesita pasarle el resultado en formato JSON:

```
//Dump results to the gridModel object
gridModel.Data = bookVolumes.Items;
//Need to pass the total results count so that we can paginate the grid
gridModel.Total = (int)bookVolumes.TotalItems;
}
//return variable is in JSON format, so that the Kendo grid understands it correctly
return Json(gridModel);
```

5.2.1.2. Añadir libro

Desde el botón existente en cada una de las líneas del resultado de la búsqueda, se lanza la acción de añadir libros, cuya entrada única es el id del libro a agregar al *e-commerce*:

```
public ActionResult Add(string id)
```

Posteriormente, se instancia el modelo pasándole el volumen de Google Books correspondiente al "id" recibido:

```
//Get volume from Google
var volume = GetVolume(id);

BooksAPIModel model = new BooksAPIModel(volume);
```

Tras comprobar que el usuario administrador que ha lanzado la operación, cuenta con permisos para administrar productos, se añade el libro mediante una subrutina llamada "AddBook" y devuelve el control de la aplicación a la vista de gestión del producto por si el administrador quisiera modificar algún dato manualmente:

```
if (!_permissionService.Authorize(StandardPermissionProvider.ManageProducts))
  RedirectToAction("AccessDenied", "Security", new { pageUrl = this.Request.RawUrl });

  //a vendor should have access only to his products
  if (_workContext.CurrentVendor != null)
  {
      model.VendorId = _workContext.CurrentVendor.Id;
  }
 //vendors cannot edit "Show on home page" property
 if (_workContext.CurrentVendor != null && model.ShowOnHomePage)
 {
      model.ShowOnHomePage = false;
 }

 //Add volume to Nop Commerce
 var productId = AddBook(model, volume).Id;
//Redirect to standard product edition page
return RedirectToAction("Edit", "Product", new { id = productId, area = "Admin" });
```

La función "AddBook", devuelve una instancia de producto con los valores obtenidos de Google Books:

```
public Core.Domain.Catalog.Product AddBook(BooksAPIModel model, Volume volume, int
storeId = 0)
        {
            //Product
            var product = InsertProduct(model, storeId);
            //Custom fields
            InsertAttributes(product, ref model);
            //Images
            var insertedPicId = InsertImages(product, volume.VolumeInfo);
            //Books categories
            InsertCategories(product, ref model, volume, insertedPicId, storeId);
            //Search engine name
            model.SeName = product.ValidateSeName(model.SeName, product.Name, true);
            _urlRecordService.SaveSlug(product, model.SeName, 0);
            return product;
        }
```

5.2.1.3 Crear una tienda

Ahora que se ha visto cómo buscar un libro y cómo añadirlo, se puede detallar cómo se crea una tienda web para una lista de categorías dadas.

La acción recibe como entrada el modelo que le envía la vista con los valores introducidos por el usuario administrador, y un booleano que nos dice si se debe seguir editando la tienda una vez creada.

```
[HttpPost, ParameterBasedOnFormName("save-continue", "continueEditing")]
public ActionResult CreateStore(BooksAPIModel model, bool continueEditing)
```

Se prepara el "proxy hub" de SignalR para enviar mensajes al navegador de cómo discurre la operación, ya que al llevar algún tiempo es conveniente mantener informado al usuario, para que no tenga la sensación de que la aplicación ha quedado bloqueada.

```
var result = myHub.Clients.All.UpdateStatus("Preparing store...");
```

Si el modelo recibido es válido (no contiene fallos de validación), se crea una instancia de la entidad "Store" para el modelo, donde volcar los valores de configuración de la tienda que introducido el usuario en el formulario.

```
if (ModelState.IsValid)
 {
     var store = model.Store.ToEntity();
     store.CompanyAddress = Request["CompanyAddress"];
     store.CompanyName = Request["CompanyName"];
     store.CompanyPhoneNumber = Request["CompanyPhoneNumber"];
     store.CompanyVat = Request["CompanyVat"];
     store.DisplayOrder = int.Parse(Request["DisplayOrder"]);
     store.Hosts = Request["Hosts"];
     store.Name = Request["Name"];
     store.SecureUrl = Request["SecureUrl"];
     store.SslEnabled = bool.Parse(Request["SslEnabled"]);
     store.Url = Request["Url"];

     //ensure we have "/" at the end
     if (!store.Url.EndsWith("/"))
         store.Url += "/";
```

A continuación se crea la tienda en nopCommerce y se le asigna el *Theme* seleccionado por el usuario administrador.

```
result = myHub.Clients.All.UpdateStatus("Creating store...");
_storeService.InsertStore(store);
```

Se obtiene el ID de la aplicación web en el servidor web de Windows, IIS, y se crea un enlace con los datos de la URL y Hosts proporcionado por el usuario administrador [16]:

```
//Add the binding to IIS server
var booksAPISettings = _settingService.LoadSetting<BooksAPISettings>();
var iisAdmin = new IISAdmin();
iisAdmin.AddBinding(_booksSettings.IISSiteId, "*", "80", store.Hosts);
```

Se ha llegado al proceso de buscar y añadir los productos a la tienda, siendo de ayuda los procesos creados para buscar y añadir un libro individualmente. Primero se almacena en una estructura de datos de tipo `List<Volumes>`, todos los libros encontrados para cada una de las categorías marcadas por el usuario, y con el límite establecido en la configuración de la extensión. Al estar ordenada la búsqueda, por defecto, por relevancia, siempre se obtienen los libros más importantes de cada categoría:

```
//Search
foreach (var item in model.MultiSelectedBISACCategories)
{
result = myHub.Clients.All.UpdateStatus("Searching for items for category " + item +
"...");
string q = "subject:" + item;
var volumesItem = GetVolumesIds(q);
if (volumesItem.Items != null)
	{
		volumes.Add(volumesItem);
		totalItems = totalItems + volumesItem.Items.Count;
	}
}


//Add
result = myHub.Clients.All.UpdateStatus("Adding books...");
var i = 1;
foreach(var volumesItem in volumes)
{
	foreach(var volume in volumesItem.Items)
	{
		var pct = i * 100 / totalItems;
		var volumeTemp = GetVolume(volume.Id);
		//We get the volume again from Google for additional data
		var addedBook = AddBook(new BooksAPIModel(volumeTemp), volumeTemp,
store.Id);
```

```
                result = myHub.Clients.All.UpdateProgress(pct);
                result = myHub.Clients.All.UpdateStatus("Added book <b>" +
addedBook.Name + "</b>");
                i++;
            }
}
```

Como se puede observar en la línea donde se rellena la variable "volumeTemp", `var volumeTemp = GetVolume(volume.Id);`, se utiliza la búsqueda individual del volumen, ya que este método de Google Books, `VolumesResource.GetRequest,` como se puede ver al entrar en la subrutina "GetVolume", proporciona más información sobre el libro que la búsqueda general `VolumesResource.ListRequest,` dentro de la subrutina "GetVolmes" (nótese la "s" del final del nombre del método). [21]

6. Pruebas

En este capítulo se trata la evaluación de la aplicación, es decir, qué testeos realizar para asegurar un correcto funcionamiento tras la puesta en producción del desarrollo. Es una fase del ciclo de vida del software donde se detectan fallas potenciales, siendo más probable que estos errores de desarrollo se encuentren a medida que crece la complejidad del producto.

Para facilitar la comprensión del trabajo realizado se tabula el detalle de cada una de las pruebas siguiendo el siguiente formato:

Tabla 27. Formato de tabulación de pruebas.

ID: identificador	Finalidad de la prueba
Condiciones	Breve descripción del estado en que se encuentra la aplicación para hacer la prueba
Resultado esperado	Breve descripción de lo que debe ocurrir para que el resultado sea satisfactorio
Requisitos relacionados	Requisitos de usuario involucrados

Tabla 28. PR-01. Comprobar validación campo Web Service Base URL para valor vacío.

ID: PR-01	Comprobar validación campo Web Service Base URL para valor vacío
Condiciones	En el formulario de configuración de BooksAPI, dejar el campo vacío y pulsar en "*Save*"
Resultado esperado	Mensaje descriptivo de error y formulario no enviado
Requisitos relacionados	RU-03

Tabla 29. PR-02. Comprobar validación campo Web Service Base URL para formato URL erróneo.

ID: PR-02	Comprobar validación campo Web Service Base URL para formato URL erróneo
Condiciones	En el formulario de configuración de BooksAPI, introducir un texto aleatorio
Resultado esperado	Mensaje de error y formulario no enviado
Requisitos relacionados	RU-03

Tabla 30. PR-03. Comprobar que valores necesarios para NopCommerce son inicializados al importar un libro.

ID: PR-03	Comprobar que valores necesarios para NopCommerce son inicializados al importar un libro
Condiciones	En el formulario de búsqueda de libros, importar un libro
Resultado esperado	Valores correctamente inicializados que de lo contrario impiden la adición del producto a

	NopCommerce. Estos campos son: • Campo precio, de ser cero, comprobar que está seleccionado el valor "*Call for Price*" • Lista de categorías al menos contiene una • Nombre de producto está siempre informado • Tipo de producto y Plantilla de producto son del tipo "*Simple product*"
Requisitos relacionados	RU-18

Tabla 31. PR-04. Comprobar que el plugin aparece en el listado de extensiones instalables y se instala.

ID: PR-04	Comprobar que el *plugin* aparece en el listado de extensiones instalables y se instala
Condiciones	Dirigirse al listado de extensiones disponibles e instalar BooksAPI
Resultado esperado	Extensión correctamente instalada
Requisitos relacionados	RU-02

Tabla 32. PR-05. Comprobar el enlace de acceso a la configuración desde el sitio de administración.

ID: PR-05	Comprobar el enlace de acceso a la configuración desde el sitio de administración
Condiciones	Dirigirse al menú desplegable de BooksAPI y seleccionar la opción "*Configure*"

Resultado esperado	El formulario de configuración de BooksAPI carga correctamente
Requisitos relacionados	RU-06

Tabla 33. PR-06. Comprobar que con la instalación se crean las etiquetas de idioma necesarias.

ID: PR-06	Comprobar que con la instalación se crean las etiquetas de idioma necesarias
Condiciones	Instalar la extensión
Resultado esperado	Tras la instalación, todas las cadenas se muestran correctamente en los formularios de configuración, creación de tienda, búsqueda de libro y propiedades particulares de producto en la página de edición de producto "*Catalog -> Manage Products -> Edit Products*", buscando un producto cualquiera y después dirigiéndose a la pestaña "*Custom values*", así como en la ficha de producto
Requisitos relacionados	RU-08

Tabla 34. PR-07. Comprobar coherencia de la cadena de búsqueda de libros introducida con los resultados obtenidos.

ID: PR-07	Comprobar coherencia de la cadena de búsqueda de libros introducida con los resultados obtenidos
Condiciones	Buscar un libro
Resultado esperado	Los libros obtenidos en la búsqueda guardan relación con la cadena buscada de autor

	o título
Requisitos relacionados	RU-09, RU-10, RU-12

Tabla 35. PR-08. Comprobar los controles de paginación de los resultados de búsqueda.

ID: PR-08	Comprobar los controles de paginación de los resultados de búsqueda
Condiciones	Buscar un libro
Resultado esperado	Cuando el número de resultados es mayor de 15, el control de paginado se muestra correctamente y permite avanzar y retroceder en los resultados
Requisitos relacionados	RU-09, RU-10, RU-11

Tabla 36. PR-09. Comprobar que los valores importados para un libro son editables y su edición se guarda correctamente.

ID: PR-09	Comprobar que los valores importados para un libro son editables y su edición se guarda correctamente
Condiciones	Agregar un libro desde la búsqueda
Resultado esperado	Los valores obtenidos de Google Books son modificables desde el formulario personalizado que carga al seleccionar la pestaña "*Custom Values*" de la pantalla de edición de producto
Requisitos relacionados	RU-16, RU-17

Tabla 37. PR-10. Pinchar en el botón "Add" de una línea de los resultados de búsqueda obtiene el libro como producto de NopCommerce.

ID: PR-10	Pinchar en el botón "*Add*" de una línea de los resultados de búsqueda obtiene el libro como producto de NopCommerce
Condiciones	Agregar un libro desde la búsqueda
Resultado esperado	El flujo de la aplicación redirige al formulario de edición de producto con la información obtenida de Google Books
Requisitos relacionados	RU-18, RU-19

Tabla 38. PR-11. Comprobar el enlace de acceso a creación de una tienda.

ID: PR-11	Comprobar el enlace de acceso a creación de una tienda
Condiciones	Navegar al menú desplegable de BooksAPI del sitio de administración de NopCommerce
Resultado esperado	Existe un enlace en el menú desplegable "*Plugins -> BooksAPI*" llamado "*Create Store*" y al pinchar carga el formulario de creación de tienda
Requisitos relacionados	RU-07

Tabla 39. PR-12. Comprobar la coherencia de las categorías escogidas en la creación de una tienda con las de los libros obtenidos.

ID: PR-12	Comprobar la coherencia de las categorías escogidas en la creación de una tienda con las de los libros obtenidos
Condiciones	Crear una tienda con determinadas categorías seleccionadas
Resultado esperado	Los libros dados de alta en la tienda corresponden a las categorías seleccionadas
Requisitos relacionados	RU-19, RU-20

7. Conclusiones

En este capítulo se explican los retos iniciales antes de abordar la realización del proyecto, qué problemas se pretenden resolver y qué resultados, a nuestro juicio, se han obtenido.

Si bien el ámbito final del proyecto se tarda más en decidirlo, sí se tenía claro que trataría sobre cómo solucionar los problemas de gestión de tiendas *online* que existen para los administradores y los manager de productos. Con la competencia creciente en este tipo de canales de venta, es cada día más importante ofrecer una gama más amplia de productos y, por tanto, hacerlos disponibles en el *e-commerce* para que los clientes puedan comprarlos a través de Internet.

Se evalúa la posibilidad de hacer un proyecto sobre un *e-commerce* sin línea de negocio definida, es decir, que vendiera todo tipo de productos, y que utilizando un *webservice*, se apoyara en API's existentes que proporcionan directorios inmensos de productos, pero finalmente se creyó mejor focalizarse en una línea de productos completa que pudiera mostrar de manera más clara el funcionamiento de un automatismo sobre el *e-commerce*.

Los retos que se presentaron fueron numerosos, aunque cabe destacar los siguientes:

- Estudio y conocimiento de una API desconocida para el desarrollador, como es Google Books API, que aunque va en la línea de muchas APIs ya conocidas por el autor, planteaba el reto de utilizar sus clases dentro de una aplicación MVC completamente desconocida para el desarrollador como es nopCommerce.
- Creación de una extensión sobre un *e-commerce* con el que nunca se ha trabajado, como es nopCommerce, que plantea a su vez dos desafíos bien diferentes:
    - conocer cómo está construida la aplicación y qué recursos públicos se tienen para poder utilizarlos en la extensión, y
    - cómo crear extensiones de manera que se evite modificar absolutamente nada del producto original. [10]

- Gestionar las operaciones de creación de la tienda de manera asíncrona e informando de las mismas al usuario desde el servidor. El autor nunca había implementado en MVC un sistema de comunicación bidireccional. Esto se ha realizado en el presente proyecto utilizando SignalR.

Todo ello ha sido superado con éxito, además habiendo ampliado los conocimientos sobre el patrón MVC, debido a que las versiones que el autor ha venido utilizando hasta ahora se limitaban a la versión 4 (nopCommerce utiliza la 5), profundizando en la gestión de servidores web IIS desde código C#, y en el amplio asunto de la estandarización documental que se ha podido estudiar brevemente gracias al estándar BISAC.

La realización de este proyecto, por tanto, ha sido un gran aliciente, ya no sólo por la satisfacción de dar por terminada la carrera, sino por la carga didáctica que ha traído consigo y que en este gremio es siempre tan necesaria. Siempre es agradable comprobar, que la titulación de ITIG ha servido sobre todo para abordar cualquier problema o requerimiento planteado, ya sea utilizando las tecnologías que ya conocidas o tecnologías nuevas, de las cuales nunca se había tenido ningún miedo en estudiar. Todo eso se debe en buena parte a los conocimientos adquiridos durante la carrera.

8. Trabajo futuro

Como se ha mencionado durante las conclusiones relativas a este proyecto, el mismo se ha centrado en desarrollar una extensión que trabaje sobre la línea de productos concreta de los libros, para poder sacar el máximo partido a Google Books y el *e-commerce* sobre el que lo se ha construido, nopCommerce, de la manera más didáctica posible.

De esta manera, uno de los trabajos que se podrían abordar en el futuro, basándose en esta idea, es la de desarrollar una extensión que apoyándose en una API de líneas generales de producto, como la que existe para Amazon, posibilite la creación de tiendas con los principales productos de cada una de las líneas seleccionadas. Es una forma de dar un valor añadido diferenciador a los distribuidores.

También se podría tener en cuenta la posibilidad de crear una extensión adicional, o modificando la que se proporciona en este proyecto, para gestionar "*fees*" por uso o comisiones por venta a través de las tiendas creadas en base a nuestra aplicación web.

Ha quedado pendiente en el presente proyecto, y sería un interesante trabajo futuro, el proveer a la extensión de una capa social que permita manejar nuestra lista de libros favoritos de Google Books desde nopCommerce, ya que la API cuenta con un servicio de autenticación por OAuth 2.0 para autorizar el login utilizando una cuenta de Gmail.

9. Bibliografía

En este capítulo se especifican las referencias consultadas para la realización del proyecto Fin de Carrera. El formato utilizado es el siguiente:

Número ordinal. Título de la referencia
Autor de la referencia (de figurar)
Dirección URL
Fecha de último acceso

[1] Open Library Books API - Openlibrary.org
Anand Chitipothu
https://openlibrary.org/dev/docs/api/books
Enero de 2015

[2] Inversion of Control Containers and the Dependency Injection pattern - Martinfowler.com
Martin Fowler
http://martinfowler.com/articles/injection.html
Febrero de 2015

[3] Model View Controller, Model View Presenter, and Model View ViewModel Design Patterns
Alexy Shelst
http://www.codeproject.com/Articles/42830/Model-View-Controller-Model-View-Presenter-and-Mod
Febrero de 2015

[4] ASP.NET Razor - Markup - W3Schools
http://www.w3schools.com/aspnet/razor_intro.asp
Febrero de 2015

[5] Overriding Admin Views and Partial Views in NopCommerce - Alex Wolf Thoughts

Alex Wolf
http://alexwolfthoughts.com/overriding-nopcommerce-admin-views-and-partial-views
Marzo de 2015

[6] Overriding Views in NopCommerce Plugin - Alex Wolf Thoughts
Alex Wolf
http://alexwolfthoughts.com/creating-menu-items-nopcommerce
Marzo de 2015

[7] Creating Menu Items in NopCommerce - Alex Wolf Thoughts
Alex Wolf
http://alexwolfthoughts.com/creating-menu-items-nopcommerce
Marzo de 2015

[8] Should web service be called inside a Model class in MVC framework? – Stackoverflow
http://stackoverflow.com/questions/9322686/should-web-service-be-called-inside-a-model-class-in-mvc-framework
Marzo de 2015

[9] 3 Ways To Display Views In Your NopCommerce Plugins (Embedded Resource, Theme Override And Custom View Engine) - ProNopCommerce.com
Woon Cherk
http://www.pronopcommerce.com/3-ways-to-display-views-in-your-nopcommerce-plugins-embedded-resource-theme-override-and-custom-view-engine
Marzo de 2015

[10] Extending entities without editing the nopCommerce (3.30) core - NopCommerce.com
Lars Niemann
http://www.nopcommerce.com/boards/t/29527/extending-entities-without-editing-the-nopcommerce-330-core.aspx

Marzo de 2015

[11] Overriding (Intercepting) NopCommerce Controllers And Actions - ProNopcommerce.com
Woon Cherk
http://www.pronopcommerce.com/overriding-intercepting-nopcommerce-controllers-and-actions?utm_source=nopcommerce&utm_campaign=forum_blog
Marzo de 2015

[12] BISAC Subject Headings, 2014 Edition
https://www.bisg.org/publications/bisac-subject-headings-2014-edition
Marzo de 2015

[13] nopCommerce REST Service Plugin - Codeplex
Woon Cherk
https://noprest.codeplex.com/
Marzo de 2015

[14] Simulated Multiple Inheritance Pattern for C# - CodeProject
David Esparza-Guerrero
http://www.codeproject.com/Articles/10072/Simulated-Multiple-Inheritance-Pattern-for-C
Abril de 2015

[15] Proposal of a modified nopCommerce architecture to extend entities - NopCommerce.com
http://www.nopcommerce.com/boards/t/30105/proposal-of-a-modified-nopcommerce-architecture-to-extend-entities.aspx
Abril de 2015

[16] Programmatically create a SubDomain - Using MVC 3
http://forums.asp.net/t/1992711.aspx?Programmatically+create+a+SubDomain+Using+MVc3

Abril de 2015

[17] Cutting Edge - Build a Progress Bar with SignalR – MSDN
Dino Esposito
https://msdn.microsoft.com/en-us/magazine/hh852586.aspx
Abril de 2015

[18] The Cutting Edge - A Context-Sensitive Progress Bar for ASP.NET MVC – MSDN
Dino Esposito
https://msdn.microsoft.com/en-us/magazine/hh580729.aspx
Abril 2015

[19] Upgrading SignalR 1.x projects to version 2 - ASP.NET
Patrick Fletcher
http://www.asp.net/signalr/overview/releases/upgrading-signalr-1x-projects-to-20
Abril de 2015

[20] Kendo UI – Telerik
http://www.telerik.com/kendo-ui
Mayo de 2015

[21] Google Books API explorer – Google
https://developers.google.com/apis-explorer/#s/books/v1/
Mayo de 2015

[22] SaaS Shopping Carts - Practical E-Commerce
Armando Roggio
http://www.practicalecommerce.com/articles/3219-8-Great-SaaS-Shopping-Carts
Junio de 2015

[23] Google Books History – Google
http://www.google.com/googlebooks/about/history.html

Junio de 2015

[24] Nuestra historia en profundidad – Google
https://www.google.com/about/company/history/?hl=es
Junio de 2015

[25] Llamada a Procedimiento Remoto – Wikipedia
https://es.wikipedia.org/wiki/Remote_Procedure_Call
Julio de 2015

[26] ASP.NET MVC Framework – Wikipedia
https://es.wikipedia.org/wiki/ASP.NET_MVC_Framework
Julio de 2015

[27] Global ecommerce trends 2015: UK leads the way in Europe and North America - Digital Strategy Consulting
http://www.digitalstrategyconsulting.com/intelligence/2015/01/global_ecommerce_trends_2015_uk_leads_the_way_in_europe_and_north_america.php
Julio de 2015

[28] El comercio electrónica avanza imparable: 7 de cada 10 internautas ya compran *online* - Puro Marketing
http://www.puromarketing.com/76/25004/comercio-electronico-avanza-imparable-cada-internautas-compra.html
Julio de 2015

[29] El mcommerce crece un 48 % en España y triplicará en crecimiento al ecommerce en 2015 - Puro Marketing
http://www.puromarketing.com/76/24799/mcommerce-crece-espana-triplicara-crecimiento-ecommerce.html
Julio de 2015

[30] Informe Sociedad de la Información en España 2014 – Telefónica
https://publiadmin.fundaciontelefonica.com/index.php/publicaciones/a

dd_descargas?tipo_fichero=pdf&idioma_fichero=_&title=Sociedad+de+la+Informaci%C3%B3n+en+Espa%C3%B1a+2014&code=323&lang=es&file=siE2014.pdf&_ga=1.183915302.1470971908.1436026818
Julio de 2015

[31] BISG – Mission & History
https://www.bisg.org/mission-history
Julio de 2015

[32] Global ecommerce trends 2015: UK leads the way in Europe and North America
http://www.digitalstrategyconsulting.com/intelligence/2015/01/global_ecommerce_trends_2015_uk_leads_the_way_in_europe_and_north_america.php
Julio de 2015

[33] Interfaz de programación de aplicaciones (definición) - Wikipedia
https://es.wikipedia.org/wiki/Interfaz_de_programaci%C3%B3n_de_aplicaciones
Julio 2015

[34] URL – Wikipedia
https://es.wikipedia.org/wiki/Localizador_de_recursos_uniforme
Julio 2015

[35] RPC – Wikipedia
https://es.wikipedia.org/wiki/RPC
Julio 2015

[36] Simple Object Access Protocol – Wikipedia
https://es.wikipedia.org/wiki/Simple_Object_Access_Protocol
Julio 2015

[37] Magento Products – Enterprise Edition
http://magento.com/products/enterprise-edition/marketing-and-merchandising
Julio 2015

10. Anexos

En el presente capítulo se aglutina el contenido adicional cuya finalidad es ser consultado en algún momento de la lectura del proyecto, o como apoyo para hacer esta memoria más consistente.

Anexo 1. Gestión del proyecto y planificación

Este apartado recoge la documentación relativa a la gestión del proyecto. A efectos de los cálculos de tiempos empleados para obtener el coste del proyecto, se va a hacer una media de horas diarias basándose en las horas de media trabajadas en mi caso. Como de lunes a viernes se ha trabajado a jornada completa, y únicamente se ha podido emplear una media de 3 horas diarias, y los fines de semana se han empleado unas 8 horas diarias, se puede decir que la media de horas al día trabajadas, incluyendo fines de semana, han sido **4,42 horas/día**.

Esta estimación se utiliza para hacer el cálculo total de horas empleadas en cada tarea, de manera que, por ejemplo, para una tarea de duración 5 días, el total en horas sería de 22,1 horas.

La gestión del proyecto se realiza en paralelo a las fases típicas del ciclo de vida del software: diseño, análisis, implementación y pruebas. Cómo se aborde cada una de estas fases depende en gran medida de la metodología adoptada.

Hay metodologías que mejoran los procesos relativos a la gestión de proyecto, como las llamadas metodologías ágiles, de las que un ejemplo es la metodología ***Scrum***[54], definida por Ikujiro Nonaka e Hirotaka Takeuchi a principios de los 80. Scrum, fija unos intervalos de tiempo predefinido para cada una de las iteraciones o *sprints*, y cada una de las iteraciones genera un entregable al cliente con una funcionalidad de las requeridas, por orden de prioridad atendiendo a la criticidad para el negocio del cliente que se estima para cada requerimiento.

[54] https://es.wikipedia.org/wiki/Scrum

1.1. Definición de tareas

En este apartado se definen las tareas en que se ha descompuesto este proyecto. Es muy importante dividir correctamente el trabajo que se va a realizar en tareas, para delimitar mejor el alcance del proyecto y dar al cliente una fecha en la que comprometerse a entregar el producto cumpliendo los requerimientos especificados en la fase de análisis.

1.1.1. Toma de requerimientos

Tarea en la que el usuario explica qué quiere de la aplicación que está solicitando.

1.1.2. Evaluación inicial

Esta tarea comprende los trabajos de análisis del estado del arte, qué *frameworks* y productos base a utilizar, y qué software se va a necesitar.

1.1.3. Propuesta de proyecto

Una vez se tiene claro de qué manera llegar a lo que pide el cliente, y se tiene claro el alcance del proyecto, se presenta una propuesta funcional y económica al cliente.

1.1.4. Diseño BBDD

Tablas y campos adicionales a incluir para los nuevos elementos necesarios.

1.1.5. Diseño interfaz

Maquetado de los elementos de la interfaz de usuario que van a intervenir, *mock-ups*, diseños conceptuales, temas (*themes*), fuentes, colores a utilizar…

1.1.6. Estudio del modelo E/R

Estudio de la BBDD existente en NopCommerce y de qué entidades se dispone para el desarrollo de la extensión.

1.1.7. Estudio arquitectura y MVC

Diseño de cómo desarrollar la extensión dentro del patrón MVC y de qué elementos se dispone en cada una de las capas de la aplicación NopCommerce.

1.1.8. Implementación del *Entity Framework*

Creación de las entidades a utilizar en el dominio de la extensión, así como sus interfaces y servicios.

1.1.9. Pruebas API Google Books

Testeo de la consola de administración de Google Books y con ejemplos disponibles en la red para idear el controlador que maneja la comunicación con la API.

1.1.10. Implementación del modelo

Desarrollo del modelo BooksAPIModel incluyendo los campos que se utilizan de Google Books.

1.1.11. Implementación del controlador

Desarrollo de las acciones que se necesitan en cada una de las llamadas desde las vistas.

1.1.12. Implementación de las vistas

Creación de las *Razor views* necesarias para presentar al usuario. Se corresponde con el desarrollo de la interfaz de usuario.

1.1.13. Documentación

Redacción de la presente memoria y recopilación de toda la información y todos los datos necesarios para la realización del proyecto.

1.1.14. Entrega

Revisión de los requisitos funcionales implementados. Preparación de la presentación del proyecto terminado.

1.2. Diagrama de Gantt

El diagrama de Gantt propuesto (ver figura 23), sirve para detallar la planificación previa al inicio del proyecto o en la fase de diseño, para exponer al cliente la dedicación en cada una de las tareas definidas, y así como las fechas en las que estimamos el proyecto finalice cada uno de sus hitos.

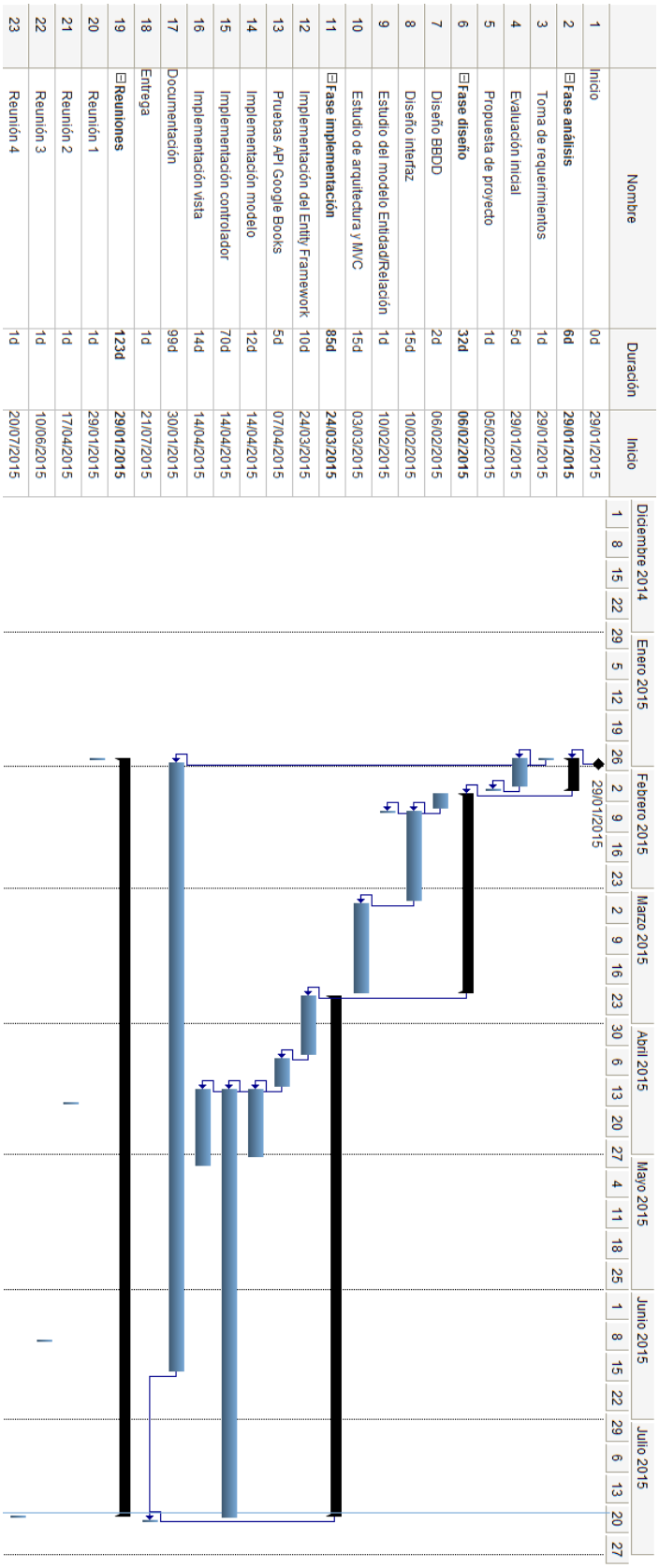

Figura 23. Diagrama de Gantt.

Anexo 2. Presupuesto

A continuación se detallan los principales costes que acarrea el proyecto, desglosados por costes del personal, de hardware, software y licencias. Todos los costes están calculados sin I.V.A.

2.1. Resumen de horas empleadas

Tabla 40. Resumen de horas empleadas

Fase	Días	Horas	Total (h)
Análisis	6	4,14	24,84
Diseño	32	4,14	132,48
Implementación	85	4,14	351,9
Total	**113**	**4,14**	**509,22**

2.2. Resumen de coste de personal

Se ha estimado el coste por hora de un ingeniero informático en 31 €/h.

Tabla 41. Resumen de coste de personal

Perfil	Horas	€/h	Total (€)
Analista	24,84	31	770,04
Diseñador	132,48	31	4.106,88
Programador	351,9	31	10.908,90
Total			**15.785,82**

2.3. Resumen de coste de Hardware

El cálculo especificado a continuación incluye la depreciación de los equipos utilizados durante el proyecto, resultando en un coste de amortización reflejado en los totales. Todos los costes están calculados sin incluir I.V.A.

Tabla 42. Resumen coste de Hardware.

Descripción	Unidades	Coste (€)	Dedicación (m)	P. Depreciación (m)	% dedicación	Amortización (€)
Ordenador AMD FX - 4170 Quad Core a 4.20 GHz	1	700	6	60	100	70
Monitor ASUS IPS 21"	1	200	6	60	100	20
Monitor HP LCD 17"	1	180	6	60	100	18
Ratón inalámbrico	1	20	6	60	100	2
Teclado	1	10	6	60	100	1
USB WiFi dongle	1	20	6	60	100	2
					Total	**113,00**

La fórmula empleada para calcular la amortización ha sido:

Dedicación (m) / Período de depreciación (m) * Coste * % dedicación/100

2.4. Resumen de coste de Software

En este apartado se detalla el gasto dedicado a la adquisición de licencias de Software para la realización del proyecto.

Tabla 43. Resumen de coste de Software

Descripción	Unidades	Coste (€)
Office 2013 Personal	1	119
VMWare Workstation 11	1	226,22
Visual Studio 2013 Express	1	0
SQL Server 2012 R2 Express	1	0
SQL Server Management Studio 2012 Ex	1	0
SourceTree	1	0
Google Chrome	1	0

Total	**345,22**

2.5. Coste total

Sumando los costes parciales anteriormente detallados se obtiene el coste total del proyecto sin incluir I.V.A.:

Tabla 44. Coste total del proyecto

Descripción	Coste (€)
Personal	15.785,82
Hardware	113
Software	345,22
Total	**16.244**

Anexo 3. Manual de instalación y configuración

En el siguiente anexo se elabora un pequeño manual de instalación para los principales módulos que intervienen en el proyecto, concretamente NopCommerce y la extensión.

3.1. NopCommerce

3.1.1. Descarga

El instalador de NopCommerce, en su versión para no ser modificado por desarrolladores, pude ser descargado desde la dirección http://www.nopcommerce.com/downloads.aspx (ver figura 24) donde automáticamente se publican las nuevas versiones que se liberan por parte del equipo de desarrollo de producto.

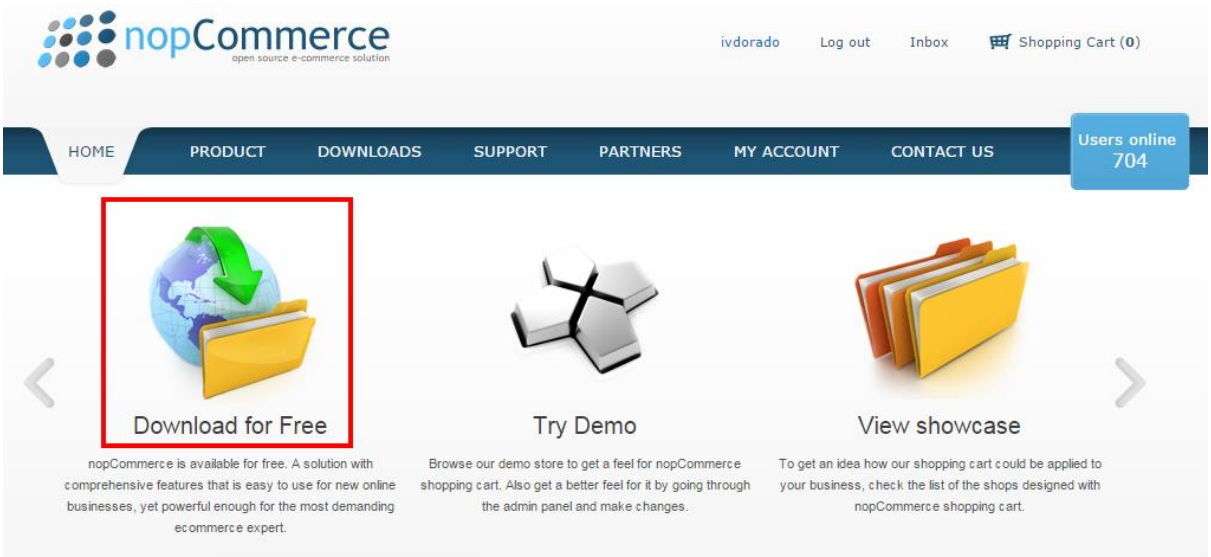

Figura 24. Página principal de NopCommerce.

También es posible instalarlo a través de la aplicación lanzada desde el "Administrador de Internet Information Server" en Windows, llamada "Instalador de plataforma web", o "*Web platform installer*":

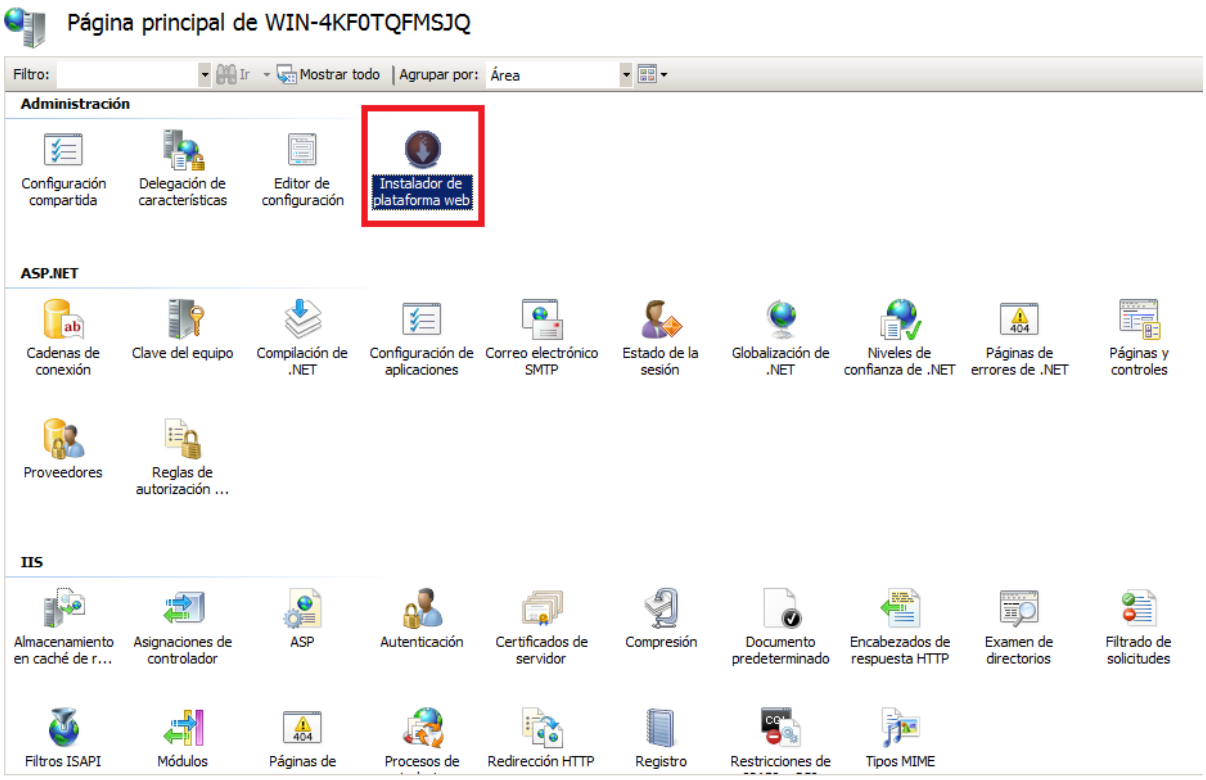

Figura 25. Panel de administración del IIS. Instalador de plataforma web.

En este manual se muestra, sin embargo, cómo hacerlo desde el instalador que se descarga desde la web de NopCommerce.

Actualmente, la versión descargable es la 3.60, con un tamaño de 28.80 MB. Es un fichero de extensión ".rar", comprimido del programa WinRar[55] (ver figura 26), y que contiene las siguientes carpetas de aplicación web:

[55] https://www.winrar.es/descargas/winrar

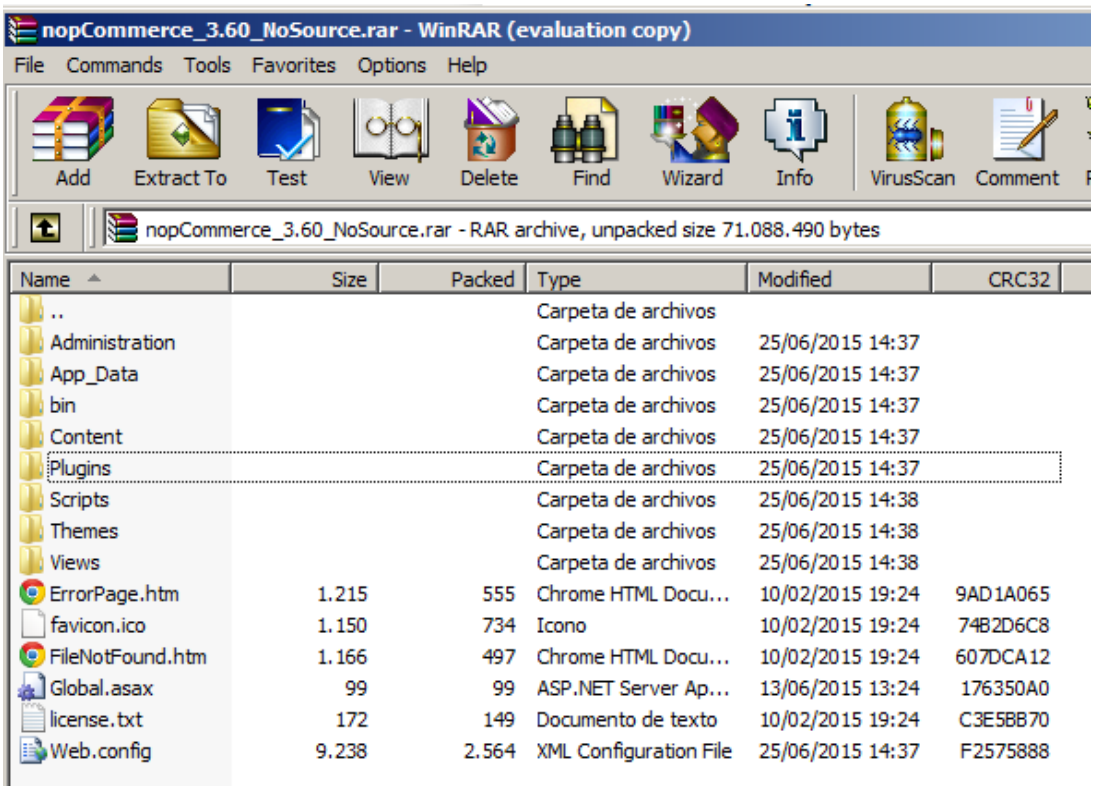

Figura 26. Contenido del fichero comprimido con la aplicación web.

Lo descomprimimos en una carpeta de nuestro equipo.

3.1.2. Configuración en servidor web IIS

A continuación se da de alta la aplicación web en el servidor web, pinchando en la carpeta de "Sitios" con el botón derecho, y seleccionando "Agregar sitio web" (ver figura 27):

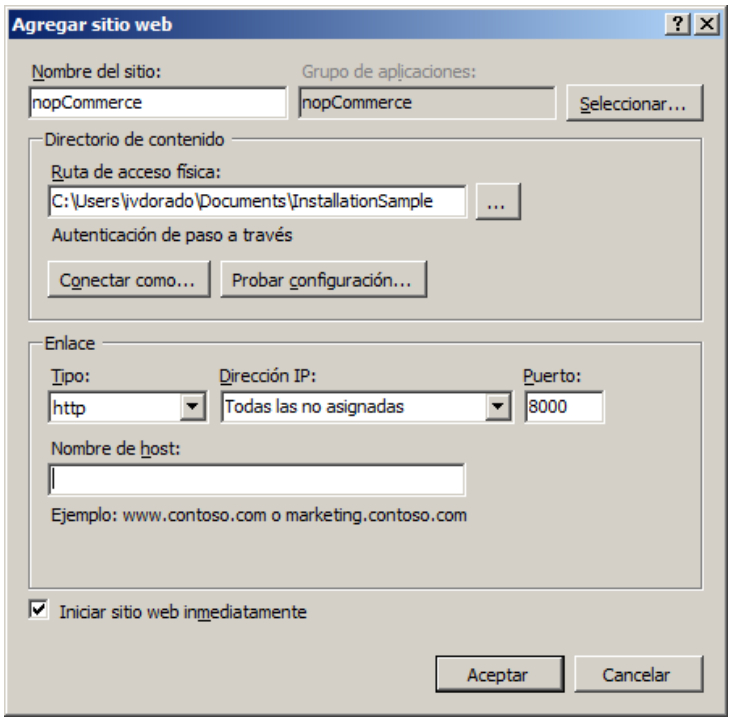

Figura 27. Agregar sitio web en el IIS.

Los datos a configurar en esta ventana son:

- Nombre identificativo del sitio. En el ejemplo "nopCommerce"
- Ruta en nuestro equipo donde se ha descomprimido la aplicación web en el punto 3.1.1.
- Número de puerto en el que se desea que escuche el servidor web para esta aplicación. En el ejemplo "8000".

Los datos no configurados, aunque dependiendo de cada instalación es posible que haya que configurar son:

- Grupo de aplicaciones: representa el hilo o proceso de Windows que va a ejecutar la aplicación web.
- Tipo: "http", sin encriptación mediante certificado de servidor, o "https" utilizando encriptación SSL mediante un certificado de servidor.
- Nombre de host: es el sitio para el que el servidor devuelve la aplicación web en las peticiones HTTP.

Al pinchar en "Aceptar", el sitio web ya está ejecutándose. Es importante anotar el "Id" del sitio (ver figura 28), ya que va a servir después para la configuración de la extensión:

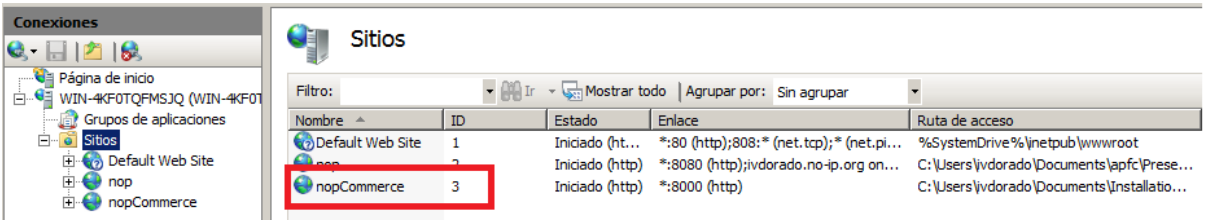

Figura 28. ID del sitio web creado.

Como no se ha especificado nombre de host, la aplicación "escucha" por defecto en "localhost" (ver figura 29):

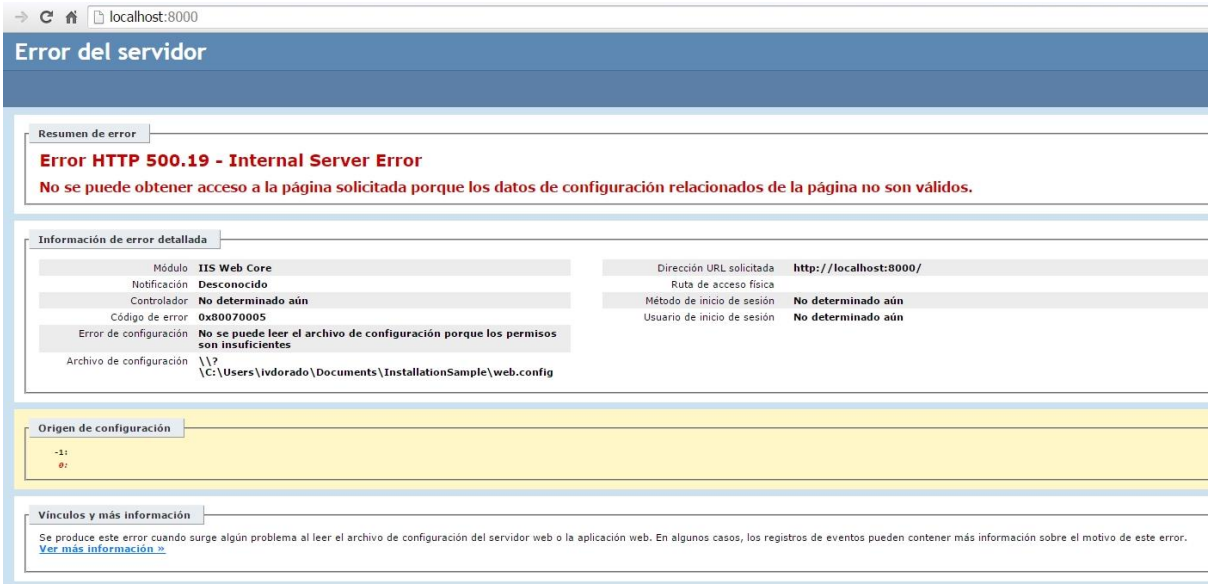

Figura 29. Error al llamar por primera vez a la aplicación web.

Como se observa, la primera vez que se accede, el servidor web avisa de que no tiene suficientes permisos en la carpeta para ejecutar la aplicación, por tanto, es necesario darle permisos de lectura y ejecución al usuario anónimo de Internet "IUSR" (ver figura 30):

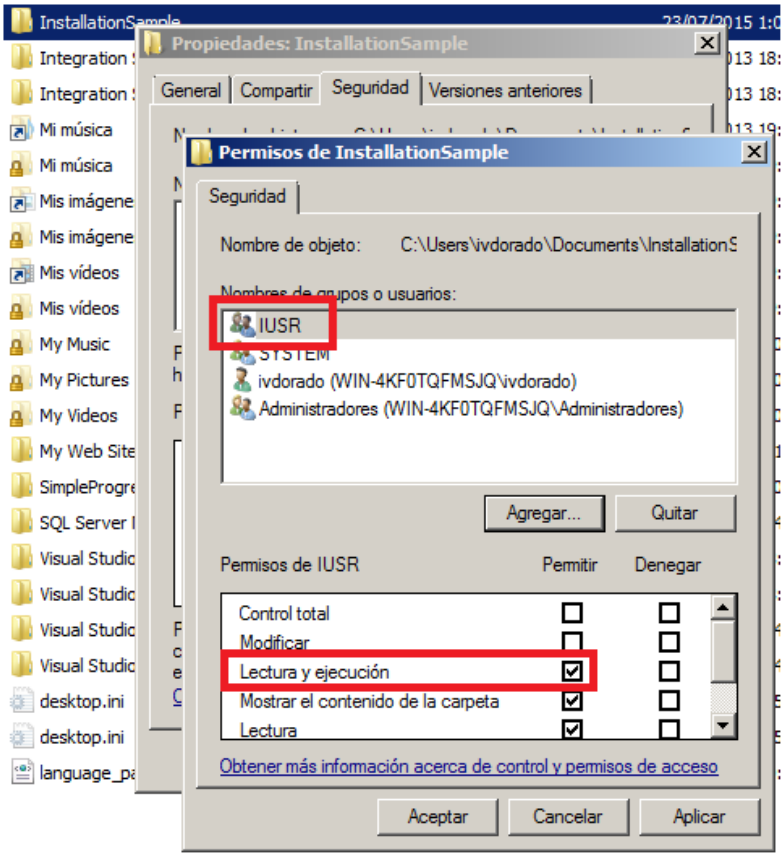

Figura 30. Permisos del directorio de NopCommerce

Al refrescar la página a continuación, la aplicación la primera vez compilará automáticamente, lo que le llevará algo más de tiempo, y posteriormente mostrará la página de instalación.

3.1.3. Instalación

La instalación de NopCommerce es extraordinariamente sencilla, ya que sólo hay que rellenar los datos de la cuenta de administrador y de SQL Server y el resto lo hará el script de instalación de NopCommerce, como se ve a continuación. Si nos dirigimos a http://localhost:8000/ se obtiene la siguiente pantalla:

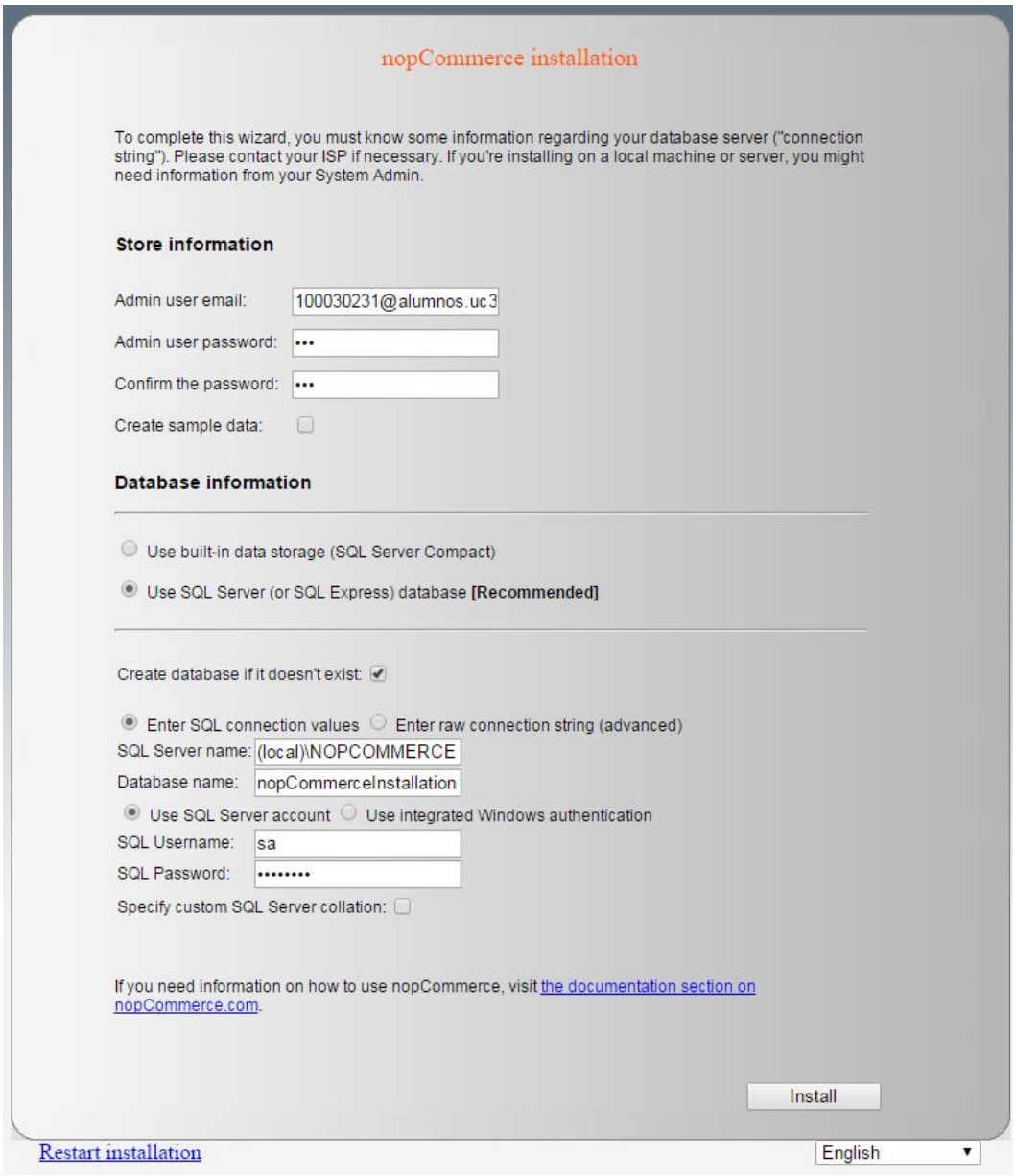

Figura 31. Pantalla de instalación de nopCommerce.

Los parámetros son auto-explicativos, y entrando un poco en detalle:

- *Admin user email y password*: el correo y la contraseña para el usuario administrador.
- *Create sample data*: opción para crear datos de ejemplo al instalar la aplicación.

- *User built-in data storage or SQL Server*: opción para utilizar una base de datos SQL Server Compact (menos escalable, para entornos reducidos, aunque no precisa de disponer de un SQL Server instalado) y SQL Server.
- *Enter SQL connection values o raw connection string*: opción para introducir los datos de instancia y base de datos de SQL Server, o el *connection string*[56], que es una única cadena donde se especifican todos los datos necesarios para establecer una conexión con SQL Server.
- Usar cuenta SQL Server o autenticación integrada de Windows: opción para utilizar un usuario ya creado en SQL server o el usuario con el que logarse en Windows.
- *Specify custom SQL Server collation*: opción para crear la base de datos con una intercalación (grupo de caracteres y definición de la sensibilidad, como mayúsculas y acentos) determinada.

Una vez ha terminado, la instalación redirige a la página de inicio de la aplicación web, lo que significa que la instalación ha terminado:

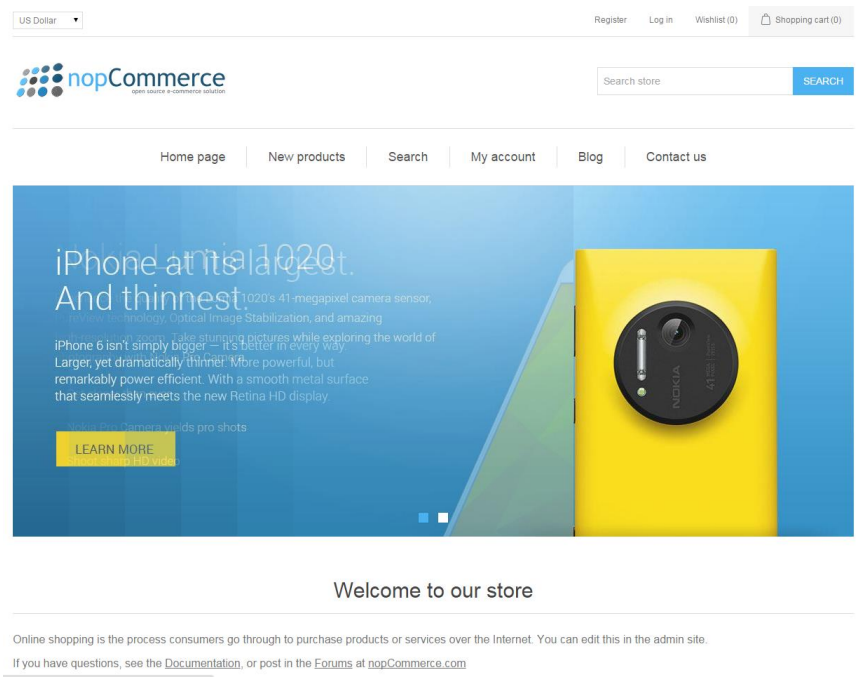

Figura 32. Página de inicio tras instalación.

[56] http://www.connectionstrings.com/sql-server/

3.2. Extensión

El sistema de NopCommerce hace muy simple la instalación de un plugin, si este se ha desarrollado específicamente como extensión de la aplicación. Por esta razón, se hizo muy complejo el desarrollo en ciertos momentos, ya que se deben personalizar ciertas áreas de la aplicación original sin tocar la misma. Es cierto que la propia aplicación NopCommerce está preparada para ello, pero esto no le resta complejidad, por ejemplo, a utilizar una vista de detalle de producto particular, en la que poder incluir los nuevos campos de producto agregados desde las clases creadas para la API de Google Books.

La instalación y configuración se hace básicamente de la siguiente manera:

- Descomprimir el fichero .zip con el código y librerías del programa.
- Instalar la extensión desde el listado de extensiones disponibles en el menú de administración de NopCommerce.
- Configurar los parámetros básicos incorporados a la extensión.

3.2.1. Descompresión del fichero Nop.Plugin.Product.BooksAPI.zip

La ruta por defecto donde depositar las carpetas con extensiones es "*Plugins*". Ahí se debe extraer directamente el contenido del fichero con cualquier programa que maneje ficheros del tipo ".zip" (ver figura 33).

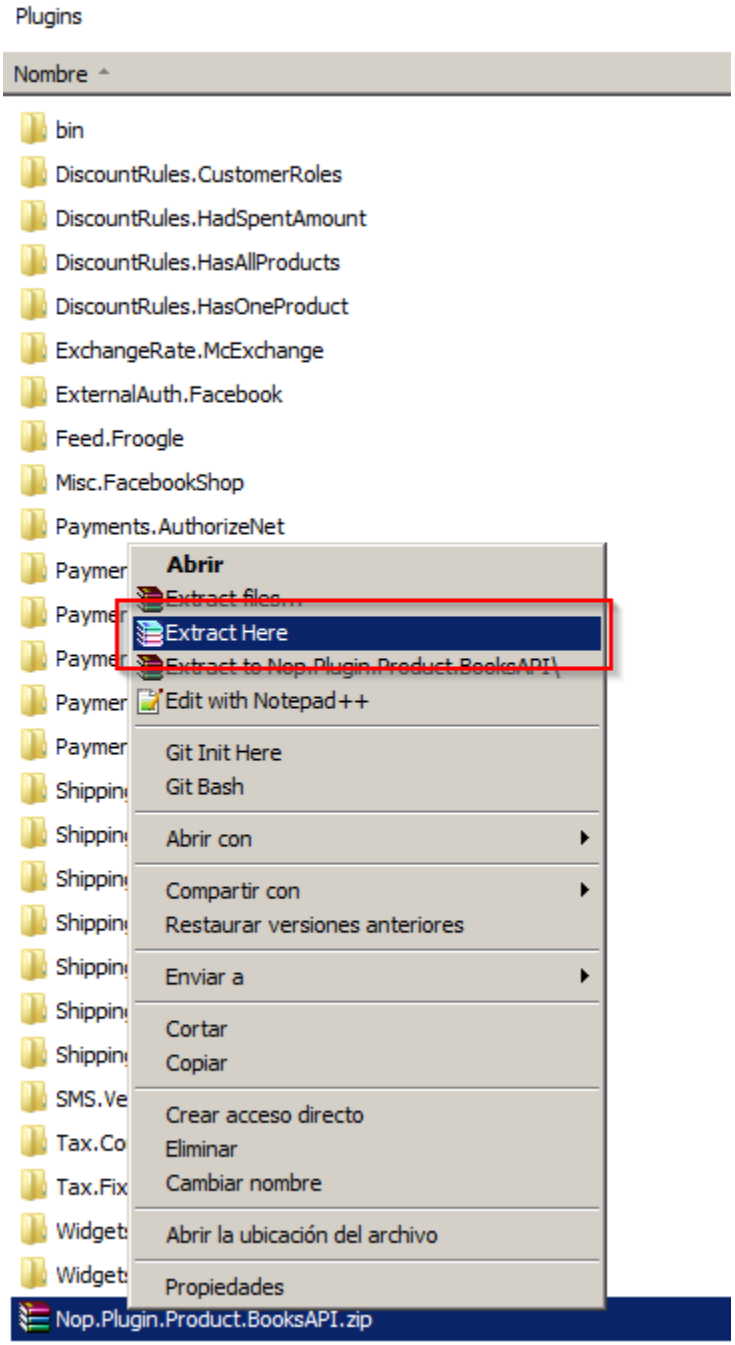

Figura 33. Extracción de la carpeta con el programa

A continuación, se detalla la carpeta que contiene la extensión (ver figura 34).

Figura 34. Carpeta una vez descomprimida.

Lo siguiente es ir al menú de administración de la extensión e instalarla.

3.2.2. Instalación de la extensión

Dirigirse al menú de "Administración" a través del link que aparece en la parte superior al logarse con el usuario administrador creado en el punto 3.1.3. Instalación.

A continuación dirigirse a "*Configuration – Plugins – Local Plugins*" (ver figura 35).

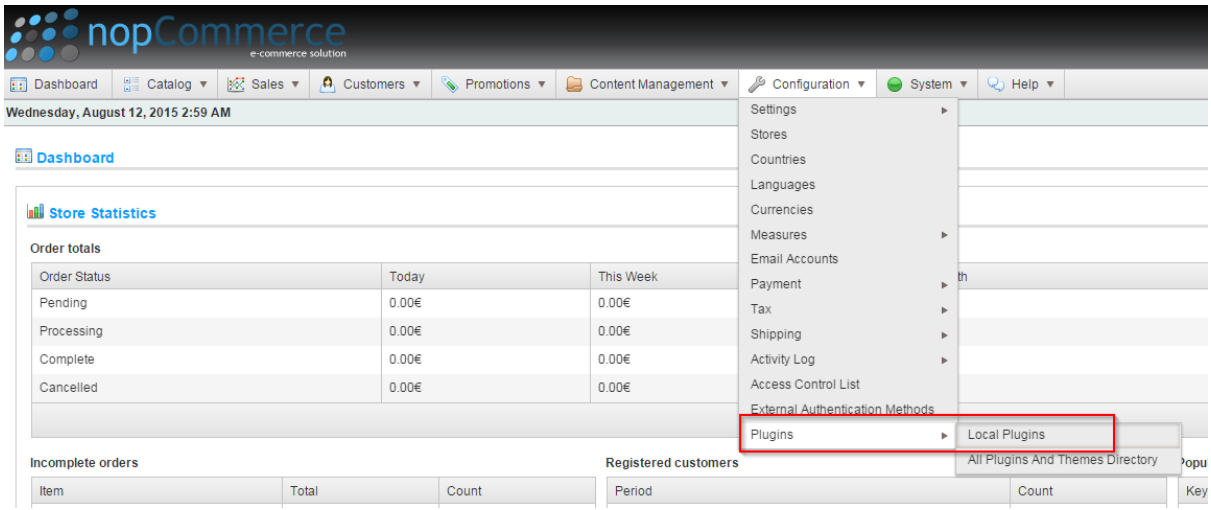

Figura 35. Menú de plugins de nopCommerce.

En la siguiente pantalla se pueden ver todas las extensiones disponibles, incluyendo la referida al presente proyecto, llamada "BooksAPI" (ver figura 36), incluyendo los botones "*Install*" and "*Edit*".

Figura 36. Extensión "BooksAPI" en el listado de extensiones.

Pinchando en el botón "*Install*" la extensión quedará instalada.

3.2.3. Configuración de la extensión

Para configurar la extensión, se ha creado un menú independiente accesible desde el sitio de administración, y en el desplegable "*Plugins*", "Books API", opción "*Configure*".

Dicha configuración (ver figura 37) es extremadamente sencilla, y consta de unos cuantos parámetros explicados a continuación:

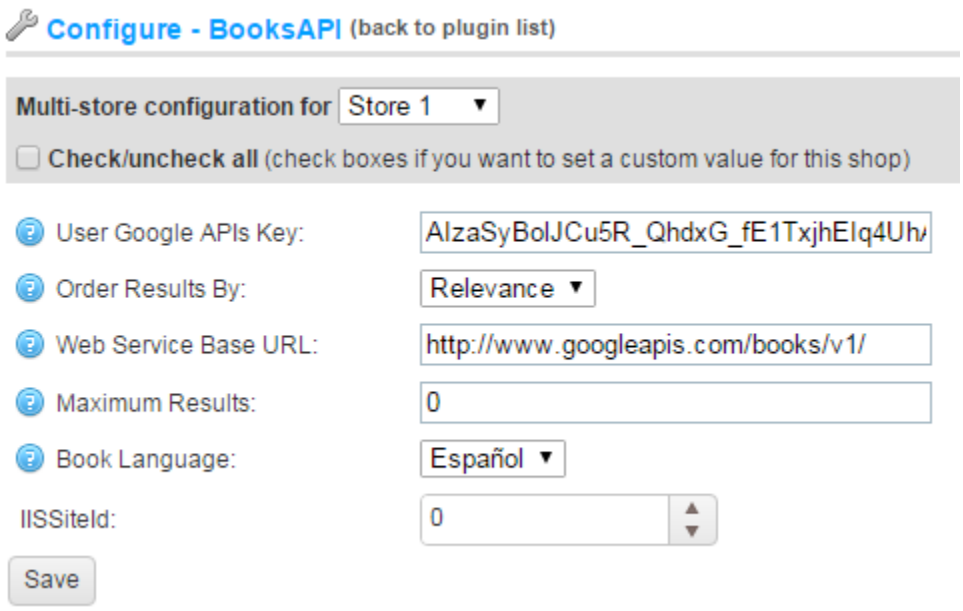

Figura 37. Pantalla de configuración de la extensión.

- ***User Google APIs Key***: es la clave que proporciona Google para conectarse con distintas APIs incluida Books API, y de esta manera utilizar software de terceros para acceder a dichas interfaces. Para ello es necesario solicitar la clave desde el *Google Developers Console* (https://console.developers.google.com/) seleccionando la API de Google Books.

- ***Order Results By*:** permite especificar qué criterio, de los proporcionados por Google, se va a utilizar para listar los resultados de la búsqueda de libros.
- ***Web Service Base URL*:** la URL donde se encuentra el servicio la API de Google Books.
- ***Maximum Results*:** número máximo de resultados que devolverá la búsqueda. Si es 0, la búsqueda devuelve todos los resultados devueltos por Google.
- ***Book Language*:** posibilita seleccionar el idioma de los libros a buscar.
- ***IIS Site Id.*:** es el ID del sitio web en el servidor IIS, anotado en el punto 3.1.2., y que la extensión utiliza para la creación de nuevas tiendas discriminando por nombre de host.

3.2.4. Ejemplo de uso

A continuación se muestra un ejemplo de uso de la extensión para proporcionar al lector el detalle de cómo utilizarla una vez instalada.

En dicha muestra, se va a:

1. crear una tienda utilizando unos parámetros de ejemplo, y seleccionando 3 categorías literarias, sobre las que basar la creación de la nueva tienda.
2. buscar e importar un volumen determinado.
3. visualización de un libro en la tienda y su pre visualización mediante el visor de Google Books.

Creación de una tienda:

Desde el menú desplegable “Plugins” del sitio de administración, dirigirse a “Books API”, “Create Store”, y allí introducir los siguientes datos (ver figura 38):

Store name: “Store 3”
Store URL: http://www.store3.com

SSL enabled: no seleccionado

HOST values: www.store3.com

Display order: 1

Company name: "Company Example"

Company address: "Company Address Example"

Company phone number: "910-000-000"

Company VAT: "0-0000000N"

Theme: seleccionado "NopRoot"

Categories: seleccionadas "ART", "BIOGRAPHY & AUTOBIOGRAPHY" y "HISTORY"

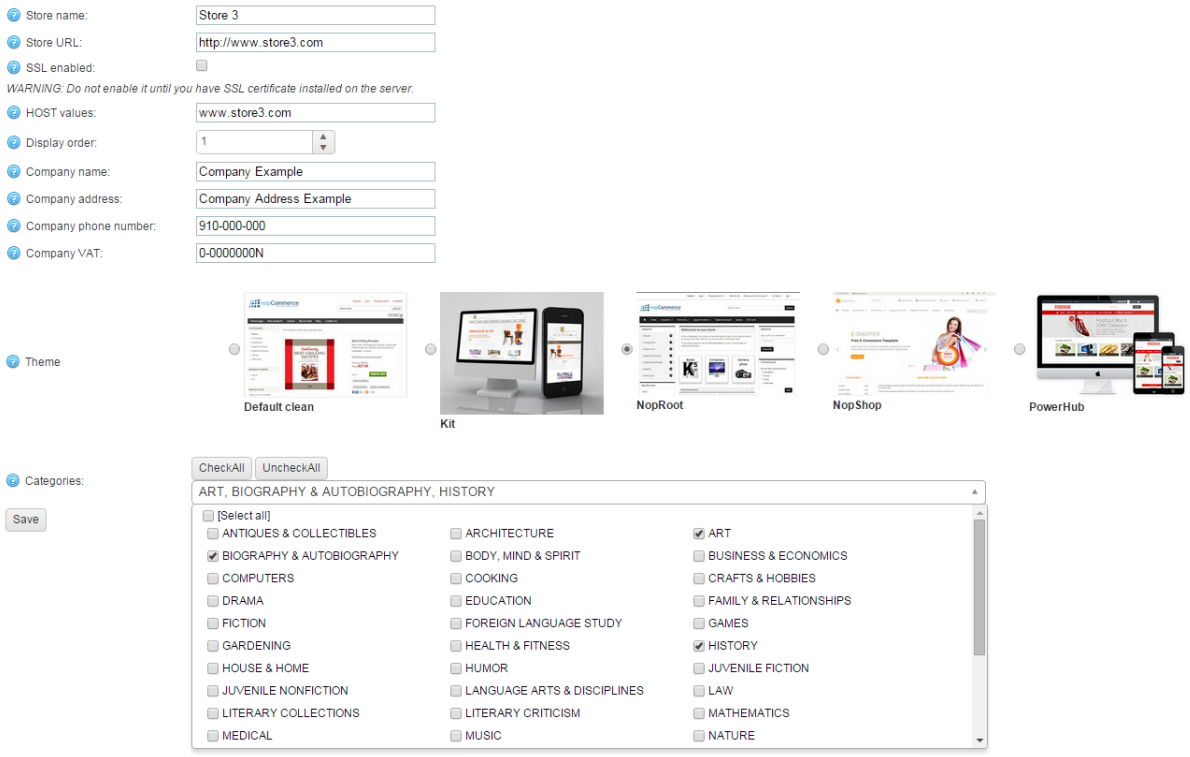

Figura 38. Pantalla de ejemplo de uso. Creación de una tienda.

A continuación, pinchar en "Save" lo que hará que comience el proceso de creación de la tienda:

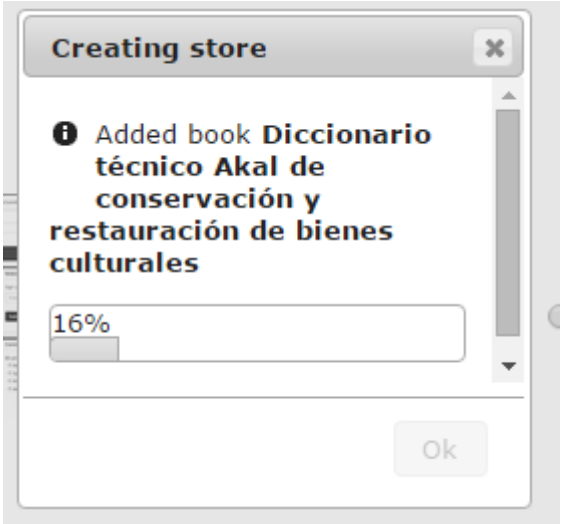

Figura 39. Proceso de creación de la tienda.

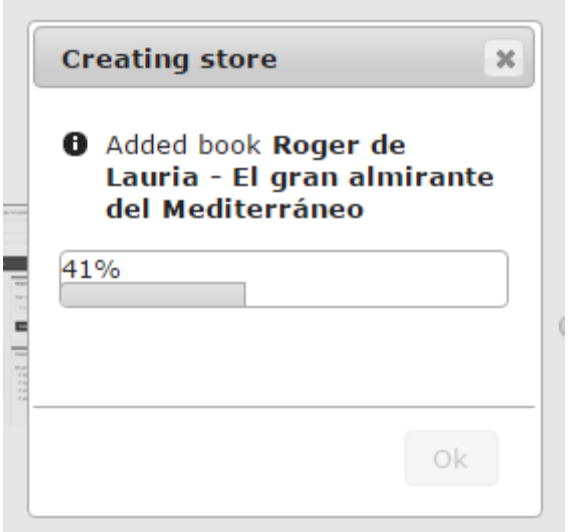

Figura 40. Proceso de creación de la tienda.

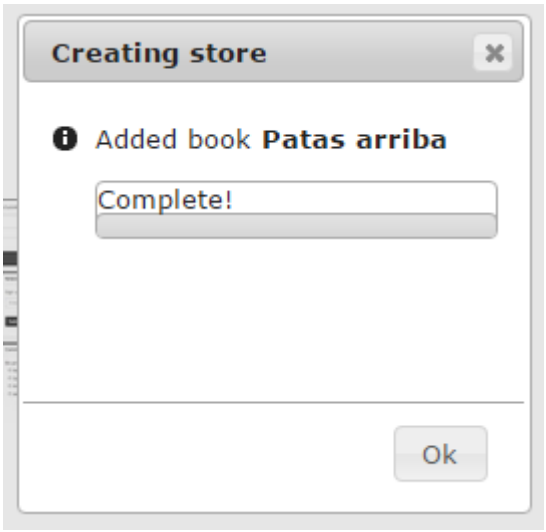

Figura 41. Proceso de creación de la tienda.

Al visualizar el texto "*Complete*!" se tiene certeza de que el proceso de importación de productos en la nueva tienda ha terminado. Pinchando en "Ok", la aplicación listará las tiendas existentes incluyendo la que se ha creado (ver figura 42):

Figura 42. Listado de tiendas.

Buscar e importar un volumen determinado:

Otra posibilidad es buscar un libro determinado, mediante título, ISBN o categoría, e importarlo de manera individual. Para ello se debe acceder a este módulo en el sitio de administración desde "*Plugins*", "Books API", "*Search book*".

Se visualiza a continuación una pantalla (ver figura 43) donde introducir los siguientes parámetros:

- ***Search*:** cadena por la que buscar el libro, que puede ser el autor, el título o alguna palabra clave relacionada con la temática del libro. A modo de ejemplo, introducir "Benedetti antología".
- **ISBN:** es el identificador único de cada volumen, por el que además poder buscar. En este ejemplo se deja vacío.
- ***Category*:** seleccionar la categoría dentro de la cual buscar el libro. Seleccionar "POETRY".

Al pinchar en "Search" se obtienen todos los resultados que envía Google:

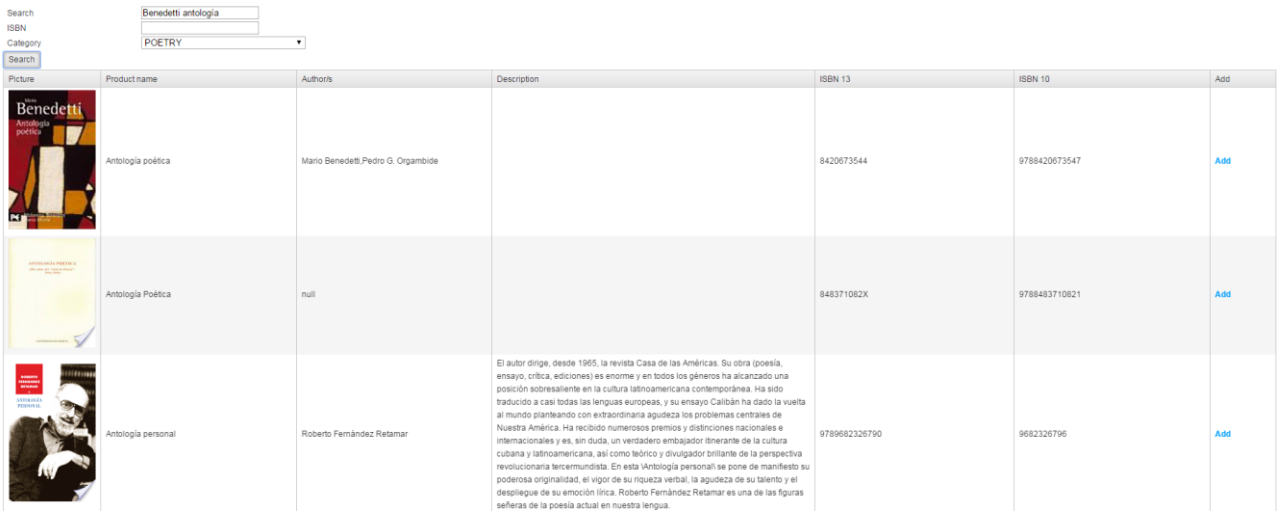

Figura 43. Resultados búsqueda libro.

Al pinchar en cualquier botón "*Add*" de cada una de las filas, el programa importa el título a la tienda y muestra la página de edición (ver figura 44) para modificar cualquiera de sus características si así se deseara.

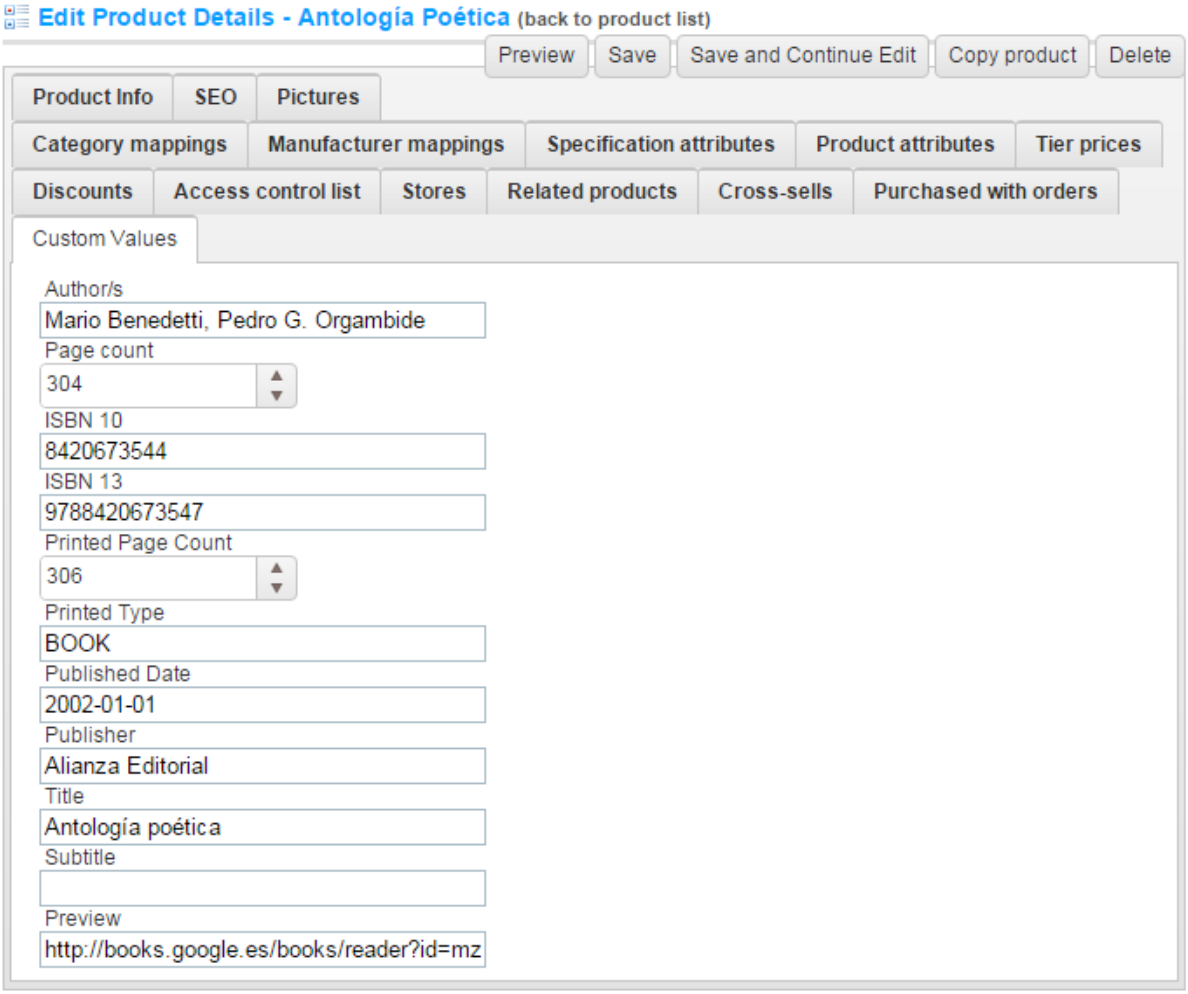

Figura 44. Página de edición de las características del libro.

Visualización de un libro en la tienda y su pre visualización mediante el visor de Google Books

Ahora que se ha agregado un libro a la tienda, ya está disponible para ser buscado y comprado (ver figura 45):

Figura 45. Ficha de producto agregado.

Pinchando en el botón “Google Preview” se obtiene, en caso de existir, una ventana con la pre visualización de algunos capítulos del libro (ver figura 46).

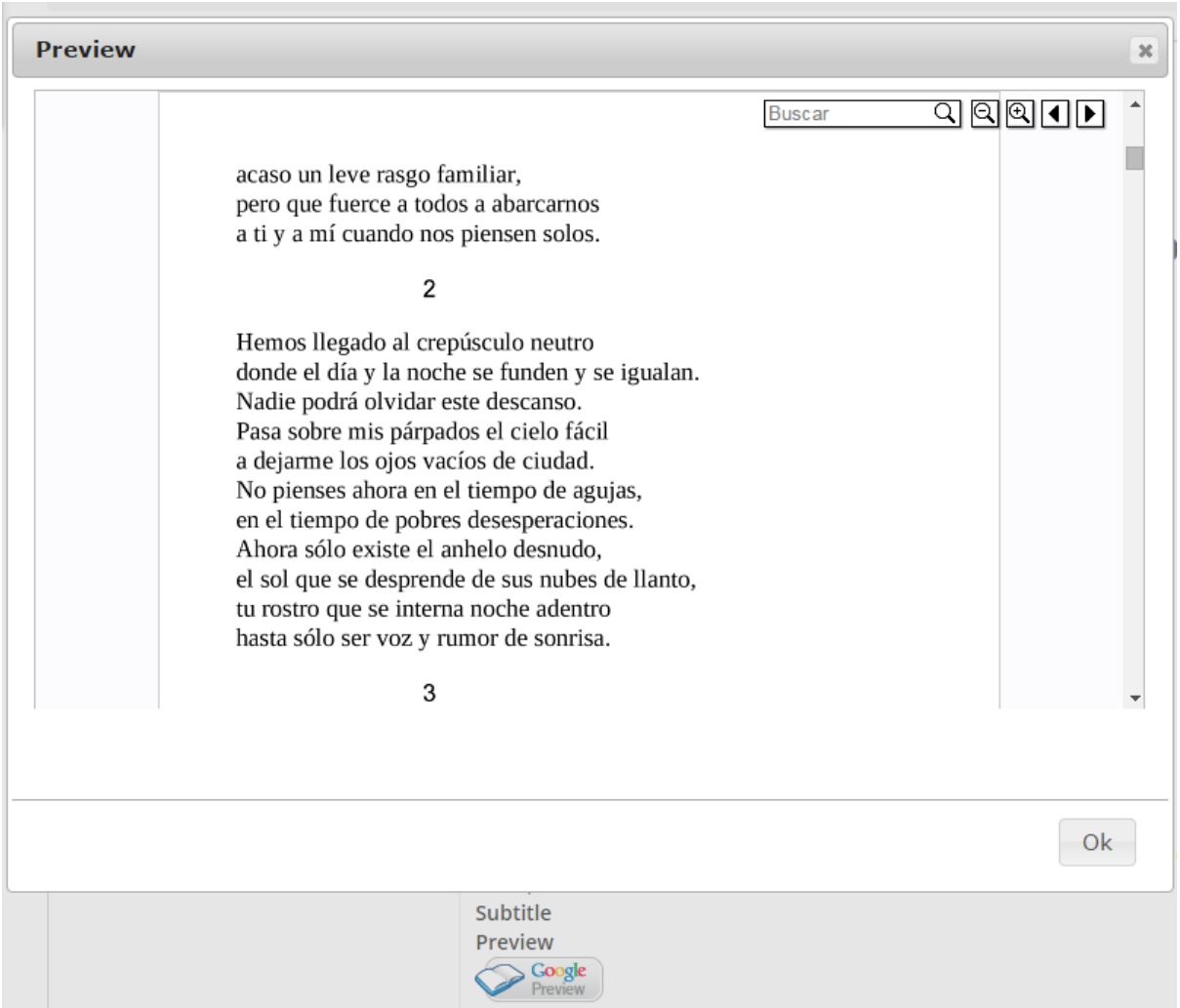

Figura 46. Pre visualización del libro agregado.